POLITECNICO DI MILANO

Facoltà di Ingegneria

Scuola di Ingegneria Industriale e dell'Informazione

Dipartimento di Elettronica, Informazione e Bioingegneria

Corso di Laurea Magistrale in

Ingegneria Informatica

Bluetooth Low Energy e Risparmio Energetico

Relatore:

ING. RAFFAELA MIRANDOLA

Correlatore:

PHD DIEGO PEREZ-PALACIN

Tesi di Laurea Magistrale di:

LORENZO PAGLIARI

Matricola n. 798273

Anno Accademico 2014-2015

POLITECNICO DI MILANO

Facoltà di Ingegneria

Scuola di Ingegneria Industriale e dell'Informazione

Dipartimento di Elettronica, Informazione e Bioingegneria

Corso di Laurea Magistrale in

Ingegneria Informatica

Bluetooth Low Energy e Risparmio Energetico

Relatore:

ING. RAFFAELA MIRANDOLA

Correlatore:

PHD DIEGO PEREZ-PALACIN

Tesi di Laurea Magistrale di:

LORENZO PAGLIARI
Matricola n. 798273

Anno Accademico 2014-2015

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Lorenzo Pagliari: *Bluetooth Low Energy e Risparmio Energetico* , Settembre 2015

DEDICHE

dedico il tutto al domo
campione indiscusso....
ovviamente xD

RINGRAZIAMENTI

Ringrazierò qualcuno ahahah....

INDICE

ELENCO DELLE FIGURE

ELENCO DELLE TABELLE

LISTINGS

ACRONIMI

IT Information Technology

QoS qualità del servizio

DVS Dynamic frequency/voltage scaling

ACPI Advanced Configuration and Power Interface

BT Bluetooth

BLE Bluetooth Low Energy

PB Probabilistic Broadcast

PE Probabilistic Edge

FF Fixed Fanout

DF Dyanmic Fanout

AL Advertising Limit

AE Advertising Event

CE Connection Event

P2P Peer-to-Peer

RGG Random Geometrig Graph

SI Suscettible – Infected

SIS Suscettible – Infected – Suscettible

SIR Suscettible – Infected – Removed

LTE Long Term Evolution

MCC Mobile Cloud Computing

NED Network Definition

SOMMARIO

Questo lavoro si posiziona nell'ambito del risparmio energetico e nell'ambito delle applicazioni mobile. Abbiamo studiato una possibile soluzione in grado di trasmettere informazioni attraverso dispositivi mobili, quando si verificano scenari nei quali le comuni reti di telecomunicazione non sono disponibili. Scenari di questo tipo possono essere causati facilmente da forti e avverse condizioni meteorologiche. Grossi nubifragi, forti nevicate o altri eventi di questo genere, possono facilmente danneggiare la struttura della rete di telecomunicazioni, causando l'inattività delle stesse. La nostra idea è stata di progettare un sistema in grado di sfruttare i comuni dispositivi elettronici più diffusi tra le persone e, tramite tecnologie equipaggiate su di essi, per trovare un modo alternativo che non si basasse sulle reti di comunicazione standard, per diffondere informazioni tra le persone. Lavorando con dispositivi mobili, è stato necessario lavorare anche sul discorso energia e consumi. Uno negli obiettivi del sistema che presenteremo in questo elaborato, è quello di regolare il carico di lavoro che si vuole assegnare al dispositivo, in modo da trovare sempre un compromesso tra prestazioni e autonomia residua della batteria. In questo elaborato presenteremo la soluzione da noi proposta, illustrando i principi degli algoritmi utilizzati per guidate la diffusione dell'informazione, i modelli usati per descrivere la rete di dispositivi mobili e la tecnologia di trasmissione utilizzata.

1 INTRODUZIONE

Intro....

1.0.1 *Struttura del documento*

Questo documento è stato strutturato così:

- Cap.1-Introduzione: $\cdots$.
- Cap.2-Stato dell'arte: $\cdots$.
- Cap.3-Introduzione del problema: $\cdots$.
- Cap.4-Progettazione logica: $\cdots$.
- Cap.5-Architettura del sistema: $\cdots$.
- Cap.6-Simulazioni e valutazione risultati: $\cdots$.
- Cap.7-Direzioni future: $\cdots$.
- Cap.8-Conclusioni: $\cdots$.

2

STATO DELL'ARTE

2.1 RISPARMIO ENERGETICO

2.1.1 *Introduzione*

Nel settore dell'Information Technology (IT) è ormai da parecchi anni che si guarda al problema dei consumi energetici e alle possibili soluzioni per ridurli. La tendenza verso cui questo settore sta andando è quella di spostare la parte di computazione e di storage sul Web, a carico a grandi aziende che offrono servizi di hosting o clouding. In generale ogni azienda che possiede almeno una server farm medio - grande, si dedica allo studio del risparmio energetico, in quando la spesa per l'energia per il ramo IT non è indifferente ai fini del budget aziendale. Sono stati fatti studi con lo scopo di trovare possibili soluzioni in grado di ridurre gli sprechi. Anche se non ce ne rendiamo conto, nei sistemi ITf vi sono molti sprechi di energia. In [25] è illustrato che una stima del costo energetico di un Data Center di Google sia nell'ordine dei milioni di dollari per anno e che il consumo energetico del settore IT a livello mondiale sia stato stimato attorno ai 40 miliardi di dollari nel 2009. Questo fa capire come con l'evolvere della tecnologia, essa diventi sempre più potente e performante, ma necessiti anche di una quantità maggiore di energia. Per questo motivo i discorsi di consumo energetico, della riduzione degli sprechi e dell'ottimizzazione nell'uso delle risorse sono divenuti sempre più importanti e oggetto di ricerca.

2.1.2 *Sistemi Self-Adaptative*

In letteratura si trovano molti studi riguardanti possibili soluzioni per una migliore gestione delle risorse e minimizzazione dello spreco di energia. Questi tipi di sistemi si chiamano Self-Adaptative, poiché riescono ad auto regolare parametri interni con l'obiettivo di consumare il minimo quantitativo di energia necessario. In applicazioni reali, bisogna però tenere conto non solo dei vincoli energetici, ma anche della qualità del servizio. E' logico pensare che un'azienda con un data center offra servi-

zi sul web, quindi queste soluzioni self-adaptative devono trovare un compromesso tra consumo di energia e qualità del servizio (QoS) da garantire. Questo trade-off è il motivo per cui, al giorno d'oggi, non si sono trovate soluzioni definitive. In letteratura vi sono molti articoli riguardanti sistemi auto adattativi con vincoli di QoS, come il [13], [24], [14] e [15]. L'idea di base è di poter gestire la quantità di risorse per offrire un determinato servizio, in modo da allocarne il minor numero possibile garantendo allo stesso tempo una qualità del servizio accettabile senza che l'utente ne percepisca un degrado. Questo tipo di adattamento è ovviamente molto dinamico e molto dipendente dalla distribuzione delle richieste che vengono fatte per quel particolare servizio. Ad esempio, in [15] è discussa la situazione di un server mail aziendale ed è presentato su grafico l'utilizzo del server mail nel tempo, nell'arco di una settimana. Dal grafico risulta che tra un giorno e il successivo, quindi di notte, vi è un uso limitato, mentre nella parte diurna dei giorni lavorativi si hanno alti livelli di utilizzo. Nel week-end invece un uso trascurabile. In questo caso tramite un monitoraggio, si può creare un sistema self-adaptive in grado di reagire in anticipo alle variazioni delle richieste in arrivo al server mail, allocando o deallocando risorse preventivamente. Più l'attività che si cerca di studiare non ha una distribuzione di richieste ben definita o è soggetta ad alta variabilità e magari a imprevedibili momenti di burst di richieste, più sarà difficile creare un sistema auto adattativo in grado di reagire preventivamente nel modo corretto. In questo caso lo studio del trade-off è più complicato e bisogna affidarsi a più euristiche. In generale, il concetto è di avere un sistema in grado di allocare solo le risorse necessarie a gestire l'attuale carico di lavoro col minor spreco di energia e garantendo una certa QoS. Il sistema deve anche poter anticipare un cambiamento nella frequenza di richieste in ingresso al sistema, allocando o deallocando risorse, sempre con l'obiettivo di avere attive solo le risorse necessarie a gestire il carico di lavoro.

Altri esempi di più recente attualità riguardano il mondo mobile. Dall'avvento dei primi smartphone, gli utenti si sono resi subito conto che uno dei problemi di questi dispositivi è l'eccessivo consumo energetico e la conseguente poca autonomia rispetto ai cellulari della precedente generazione. Infatti, le case produttrici implementano su tutti i loro dispositivi mobili moduli di risparmio energia, in grado di agire sul sistema in base all'impostazione scelta. Anche se le impostazioni variano

da brand a brand, in generale si può scegliere tra una modalità normale, senza risparmio energetico, una modalità di semi risparmio energetico e una o due modalità di super risparmio energetico in cui viene disabilitato praticamente ogni tipo di servizio non essenziale, lasciando al dispositivo le sue funzioni core e niente altro. Per quanto riguarda i dispositivi mobili ci si sta muovendo verso il Mobile Cloud Computing (MCC), come discusso in [16] . Data la potenza di elaborazione con cui sono equipaggiati i dispositivi mobili oggi giorno, essi sono a tutti gli effetti dei computer in miniatura e quindi possono fare tutto; a prova del sempre crescente numero di applicazioni create e immesse sul mercato. Tutto ciò si traduce in un consumo energetico per sostenere l'esecuzione di tutte queste applicazioni e in un problema d'immagazzinamento dati. Tramite il Mobile Cloud Computing, i dispositivi mobili posso scaricare il peso dell'elaborazione di certi dati e la memorizzazione dei file direttamente nel Cloud, con il risultato di risparmiare fino al 25% di batteria e molto spazio di memorizzazione.

Un altro ambito in cui è molto importante lo studio e l'ottimizzazione del consumo energetico è quello dell'hardware. Ad esempio nei microprocessori dei computer, ma non solo, sono presenti moduli in grado di monitorare lo stato del chip, la frequenza di lavoro e altri parametri. Nello specifico il modulo Dynamic frequency/voltage scaling (DVS) permette di regolare il voltaggio e quindi la frequenza di lavoro del processore in maniera dinamica, senza dover spegnere il componente. In [24] sono presentate le specifiche dell'Advanced Configuration and Power Interface (ACPI) e proposte come open standard per la gestione della configurazione e dell'energia di sistemi singoli o insieme di sistemi.

2.2 BLUETOOTH 4.0 LOW ENERGY

2.2.1 *Introduzione*

La tecnologia Bluetooth Low Energy (BLE), nome in codice Seattle, è una tecnologia wireless ed è stata rilasciata a metà 2010, col rilascio delle le specifiche relative allo standard versione 4.0; ora alla versione 4.2 [8] . Dato un sempre maggior impiego dei dispositivi Bluetooth in svariati ambiti come l'IT, moduli di rilevamento e sensori, healtcare [9] e altri ancora, questa tecnologia è stata progettata per risolvere uno dei punti deboli delle

versioni precedenti: l'alto consumo energetico richiesto. Sulla tecnologia Bluetooth Low Energy, l'azienda lancia la gamma di prodotti denominati Bluetooth Smart identificando una serie di mercati in cui è richiesta una tecnologia a basso consumo energetico come le nascenti strutture delle smart home, healtcare, sport & fitness. Questa tecnologia è stata progettata e migliorata per tutti questi dispostivi che utilizzano la batteria "a bottone" infatti, alcuni vantaggi presentati dall'azienda sono:

- Basso consumo energetico.
- Possibile autonomia di mesi o anni per sistemi alimentati da batterie a bottone.
- Ampia compatibilità con molti smartphone, computer e tablet.

Sempre nel 2010 si venne a creare un problema di retro compatibilità dei dispositivi Bluetooth Smart con tutti i dispositivi equipaggiati con tecnologie Bluetooth Classic (il Bluetooth Classic è tecnologia dalla versione 1 fino a prima della versione 4). E' risultato che la tecnologia Smart era completamente non compatibile con i dispositivi Bluetooth Classic. Per questo motivo nel 2011 l'azienda cambiò il logo del marchio Bluetooth Smart [7], dividendolo in:

- Bluetooth Smart.
- Bluetooth Smart Ready.

L'azienda creò una versione parallela alla Smart, chiamata Smart Ready che implementa l'architettura del Bluetooth 4.0, quindi in grado di comunicare con il Bluetooth sia Smart sia Smart Ready, ma che è anche in grado di comunicare con il Bluetooth Classic, inserendo quell'elemento di retro compatibilità che mancava. Tutti i dispositivi come smartphone, computer e tablet utilizzano tecnologia Bluetooth Smart Ready, mentre dispositivi progettati per funzionare con batterie a bottone o disponibilità energetiche limitate sono equipaggiate solamente con la tecnologia Bluetooth Smart. In figura 2.1 è rappresentato lo schema di compatibilità, mentre in figura 2.2 è riportata la tabella di compatibilità pubblicata dall'azienda.

Tutti gli smartphone, dal 2010 fino ad oggi sono stati equipaggiati con la tecnologia Low Energy Smart Ready. Si è, infatti, costatato un forte incremento nel mercato, di dispositivi che sfruttano tale tecnologia, come i bracciali con tecnologia fitbit,

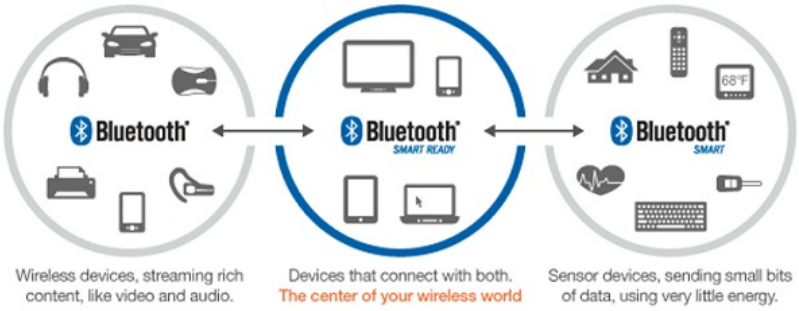


Figura 2.1: Schema di compatibilità tra dispositivi Bluetooth Classic, Smart e Smart Ready.

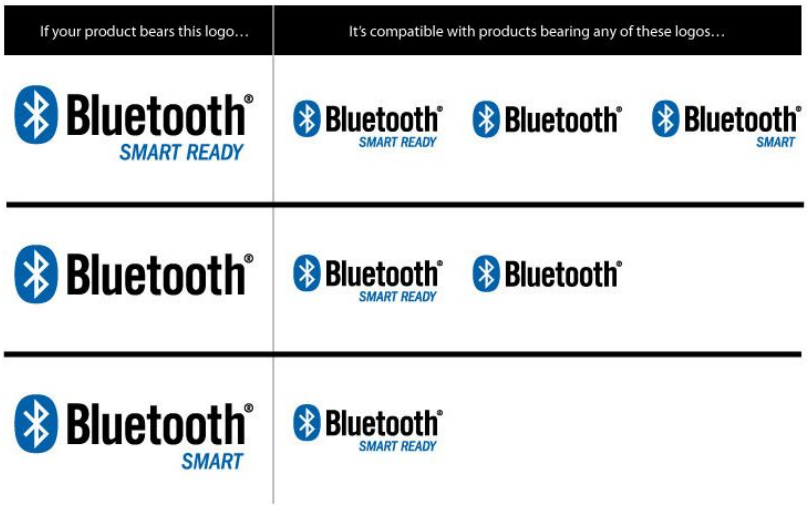


Figura 2.2: Schema di compatibilità tra tecnologie Bluetooth

la serie degli i-Watch e gli smartwatch di Android che rendono l'orologio una vera e propria estensione del telefono. Tutti questi dispositivi sfruttano la tecnologia BLE rendendo gestibile l'uso di così tante periferiche Bluetooth senza avere drammatiche conseguenze sulla durata delle batterie. Ora l'azienda Bluetooth SIG sta già lavorando a una nuova versione del BLE, la versione 4.1. Essa mira a ridurre alcuni conflitti di banda con la tecnologia Long Term Evolution (LTE). Infatti, la 4.0 si trova tra le bande 40 e 41 della tecnologia LTE. Per risolvere eventuali conflitti, la nuova versione 4.1 effettua un controllo di banda prima di iniziare le trasmissioni. Uno dei primi smatphone a essere equipaggiato con la nuova versione 4.1 del BLE è stato il Google Nexus 6.

2.2.2 *Caratteristiche generali*

In questa sezione presentiamo, in via generale, alcune caratteristiche del Bluetooth 4.0 Low Energy, anche con qualche comparazione col predecessore Bluetooth Classic per valutare i miglioramenti e i vantaggi che questa nuova tecnologia offre.

2.2.2.1 *Consumo energia*

La tecnologia Bluetooth Low Energy (BLE) presenta un consumo ridotto di energia rispetto alla versione precedente, il Bluetooth Classic. Per il Bluetooth Classic vi sono tre classi distinte in cui sono categorizzati i dispositivi, con le relative distanze raggiunte e i consumi associati, mentre per il Bluetooth Low Energy, l'azienda Bluetooth SIG non ha specificato alcun valore massimo di distanza o consumi e quindi il tutto è lasciato come libera scelta ai produttori dei trasmettitori. Questo si traduce ovviamente in una stima per il calcolo del consumo medio di questa nuova tecnologia. Nella tabella 2.1 sono riportati i valori di consumi energetici e distanze, relativi al Bluetooth Classic [6], mentre nella tabella 2.2 sono riportati i valori di consumo massimo e minimo all'output, per il Bluetooth Low Energy dichiarati nelle specifiche ufficiali [8] e il valore stimato medio di distanza raggiungibile presentato in [2]. Come si può vedere il consumo di energia del Low Energy è ridotto di almeno un ordine di grandezza rispetto al precedente standard Classic.

Come discusso in [11], il consumo molto basso da parte del Low Energy, permette a dispositivi alimentati con batterie a bottone, un ciclo di vita che varia tra i 2 giorni e i 14 anni. Inoltre rispetto allo standard Classic, che consente un massimo di 7 dispositivi slave per ogni master, la tecnologia Low Energy offre più flessibilità rendendo questo valore dipendente dall'applicazione e può variare tra 2 e 5.917 slave per master. Riguardo la distanza di trasmissione del BLE, è stato stimato essere in media attorno ai 50 metri.

2.2.2.2 *Link Layer*

Come descritto nelle specifiche ufficiali del Bluetooth 4.0[8], l'operatività del Link Layer può essere descritta dai seguenti stati:

- Stato di Standby.

	POTENZA (mW)	POTENZA (dBm)	DISTANZA (m)
Classe 1	100	20	~100
Classe 2	2.5	4	~10
Classe 3	1	0	~1

Tabella 2.1: Caratteristiche Bluetooth Classic.

POTENZA MASSIMA	POTENZA MININIMA	DISTANZA
10 mW (10 dBm)	0.01 mW (-20 dBm)	~50 m

Tabella 2.2: Caratteristiche Bluetooth Low Energy.

- Stato di Advertising.
- Stato di Scanning.
- Stato di Initiating.
- Stato di Connection.

Questi rappresentano i nodi dell'automa a stati finiti che modella il comportamento del Link Layer. Ogni macchina a stati può essere in un solo stato alla volta, ma un Link Layer può avere più istanze di macchina a stati, se il suo hardware lo consente. È necessario però che almeno una delle sue macchine sia in grado di entrare nello stato di Advertising o nello stato di Scanning. Nel caso di macchine a stati multiple, vi sono restrizioni sulle combinazioni possibili di stati attivi tra tutte le macchine a stati; per ulteriori dettagli si faccia riferimento alla documentazione ufficiale [8].

Lo stato di Standby è uno stato in cui il dispositivo non trasmette e non riceve alcun pacchetto. Questo stato è raggiungibile da qualsiasi altro stato.

Lo stato di Advertising, trasmetterà pacchetti di advertising e, se possibile, ascolterà e risponderà a quei pacchetti trasmessi in risposta al suo pacchetto di advertising. Dispositivi nello stato di Advertising sono chiamati advertiser e lo stato di Advertising è raggiungibile dallo stato di Standby.

Lo stato di Scanning è uno stato di osservazione. Un dispositivo nello stato di Scanning rimane in ascolto per i pacchetti di advertising. Un dispositivo nello stato di Scanning è chiamato

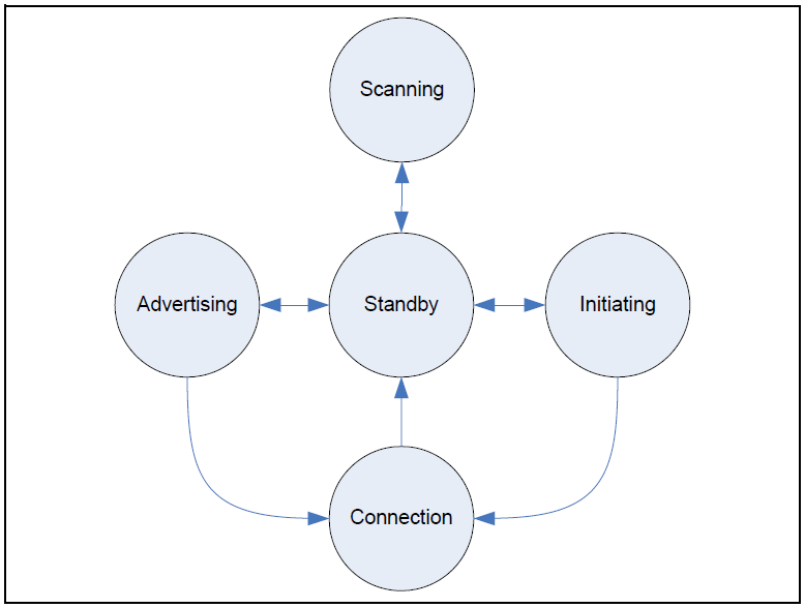


Figura 2.3: Schema di compatibilità tra dispositivi Bluetooth Classic, Smart e Smart Ready.

scanner e lo stato di Scanning è raggiungibile solo dallo stato di Standby.

Un Link Layer nello stato di Initiating rimarrà in ascolto per pacchetti di advertising trasmessi da determinati dispositivi e risponderà a determinati pacchetti se ha l'intenzione di aprire una connessione con quel dispositivo mittente. Un dispositivo in stato di Initiating è chiamato initiator. Lo stato di Initiating è raggiungibile dallo stato di Standby.

Lo stato di Connection può essere raggiunto sia dallo stato di Advertising sia dallo stato di Initiating. Lo stato di Connection si divide a sua volta in due ruoli:

- Ruolo Master.
- Ruolo Slave.

Se raggiunto dallo stato di Advertising si entrerà nello stato di Connection col ruolo di Slave, mentre se si raggiunge lo stato di Connection dallo stato di Initiating si entra nello stato Connection col ruolo di Master.Un singolo dispositivo col ruolo di Slave può comunicare con un solo Master alla volta.

In figura 2.3 è riportato lo schema della macchina a stati del Link Layer.

2.3 RETI PERR-TO-PEER

2.3.1 *Introduzione*

La nostra ricerca si è orientata su modelli di reti in grado di rappresentare reti Peer-to-Peer (P2P), chiamate anche Peer2Peer. Una rete P2P è una rete in cui non vi è una struttura gerarchica e ogni nodo è considerato allo stesso livello di tutti gli altri. Senza la presenza di nodi più importanti o nodi rappresentati centri di conoscenza della rete, nessun nodo può avere una visione completa della rete stessa e non può sapere se la porzione di rete da lui vista rappresenta l'intera rete. Ogni nodo quindi, ha una visione parziale e locale dell'overlay della rete. Le topologie di rete sono rappresentate da grafi bidirezionali, poiché i canali di comunicazione tra i nodi di una rete P2P per le situazioni analizzate, sono bidirezionali. Il modello di rete P2P è un'architettura logica di una rete di nodi paritari, senza alcuna struttura Client-Server fissi [*https://it.wikipedia.org/wiki/Peer-to-peer* da cambiare con il riferimento al libro !!]; ogni nodo è paritario a tutti gli altri, infatti, ogni nodo è chiamato peer. La struttura Client-Server è creata solo nel momento di dover instaurare una connessione tra due nodi, ma più che Client-Server, sarebbe più corretto definirla come Mittente-Destinatario perché non vi sono compiti o azioni predefinite e pre-allocate nelle due parti. Può essere quindi definita una struttura particolare della struttura generica Client-Server.

La principale applicazione di questo modello di rete è stata ed è tuttora quella della condivisione dei file (file sharing), per la quale sono nati tanti sistemi quali Gnutella, FastTrak, Napster, eMule, la rete Torrent, Freenet etc.

Alcune caratteristiche delle reti Peer-to-Peer:

- *Basso costo d'implementazione*: non è richiesto l'uso di potenti macchine Server, ma è sufficiente che ogni peer possa sostenere le transazioni dell'utente locale e degli altri peer che vogliono connettersi a lui.
- *Amministrazione decentralizzata*: non vi è un server centralizzato di stoccaggio delle informazioni, ma le informazioni sono in possesso dei singoli utenti, localmente, che poi mettono a disposizione della rete.
- *Maggiore velocità di trasmissione*: non avendo un Server centralizzato cui tutti i Client si devono connettere per avere

un'informazione, causando un calo nella velocità di trasferimento, in un'architettura P2P l'informazione può essere reperita anche da più nodi contemporaneamente, infatti, è possibile reperire parti diverse della stessa informazione da nodi diversi e alla fine riassemblare il tutto, col grosso vantaggio di poter avere l'informazione in tempi brevi.

- *Sicurezza*: senza la presenza di Server centralizzati, ogni nodo deve garantire per se e per i contenuti che distribuisce, inoltre è anche esposto a ciò che riceve dalla rete che non è controllato da nessuna terza parte.

2.3.2 *Modelli di Rete*

Poiché abbiamo preso in considerazione l'uso dei grafi, introduciamo due parametri riguardanti i grafi che ne descrivono alcuni aspetti. Essi sono la *edge dependency* e *la degree variance*.

- ***Edge Dependency***: anche chiamata interdipendenza tra archi. Dato un grafo come in figura 2.4, la edge dependency è definita come la probabilità subordinata che esista un arco tra gli stati S_i~S_j, sapendo che esiste un arco tra gli stati S_i~S_k e tra S_k~S_j. Formalmente, questa probabilità si esprime così $P(S_i \sim S_j \mid S_i \sim S_k , S_k \sim S_j)$ e si può considerare alta se è maggiore della sua probabilità semplice $P(S_i \sim S_j)$.
- ***Degree variance***: con degree variance s'intende la varianza del grado dei nodi del grafo.

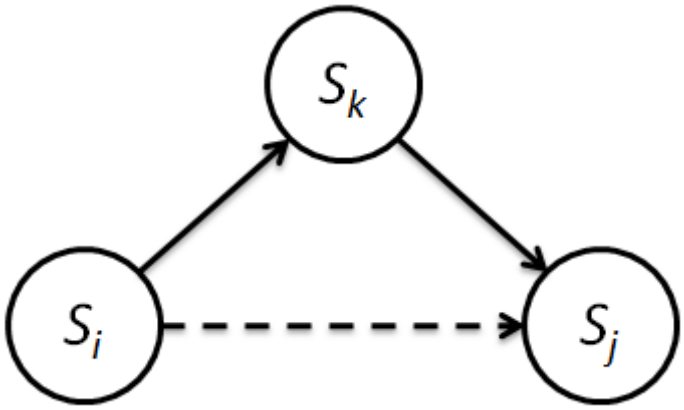


Figura 2.4: Esempio di edge dependency.

I tre principali modelli di reti presentati in [26] per la costruzione di reti P2P casuali, sono:

- Bernoulli Graph.
- Random Geometric Graph.
- Scale-free Graph.

2.3.2.1 *Bernoulli Graph*

Un Bernoulli Graph $\beta(N,p_N)$ è un grafo bidirezionale con N nodi nel quale le connessioni hanno una probabilità di esistere pari a p_N. Per ogni coppia di nodi vi è una probabilità pN che il link tra loro esista, indipendentemente da ogni altra connessione. In figura 2.5 è riportato un esempio. Un Bernoulli Graph presenta bassa edge dependency e bassa degree variance.

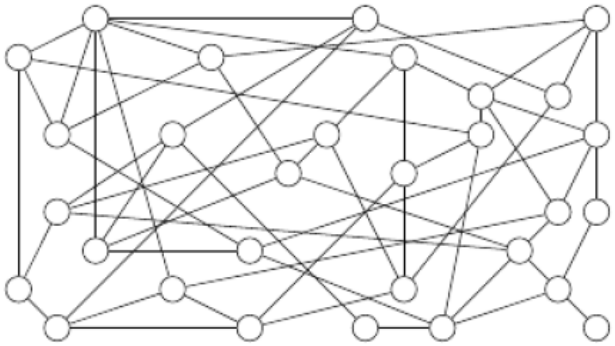


Figura 2.5: Esempio di Bernoulli graph.

2.3.2.2 *Random Geometric Graph*

Un Random Geometrig Graph (RGG) $G(N,\rho)$ è un grafo bidirezionale casuale inserito in un'area limitata. Il grafo è generato disponendo in maniera casuale e uniforme gli N nodi all'interno dell'area. Poi due nodi sono connessi se essi tra loro se essi si trovano a una distanza geometrica pari o inferiore a ρ. In figura 2.6 è riportato un esempio. E' facile notare come reti di questo tipo siano estremamente adatte alla rappresentazione di reti wireless, caratterizzate dalla distanza fisica tra i nodi e da un valore soglia ρ entro il quale è possibile eseguire trasmissioni. Un RGG presenta un'alta edge dependency, dovuta all'importanza della vicinanza fisica tra i nodi, e una bassa degree variance.

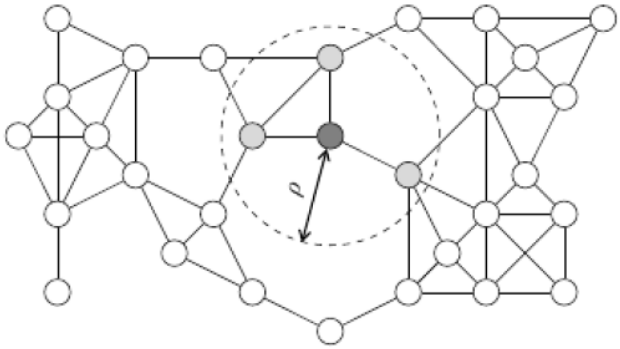


Figura 2.6: Esempio di Random Geometric graph.

2.3.2.3 *Scale-free Graph*

Un Scale-free Graph *S(N,m)* è un grafo bidirezionale con una bassa edge dependency e un'alta degree variance, dovuta alle sue connessioni distribuite secondo legge potenza. Reti di questo tipo sono generate partendo con un set di nodi m_0, poi a ogni ciclo si aggiunge un nodo e si collegano i suoi m archi ad altri nodi già presenti nel grafo. La probabilità che un nuovo nodo sia collegato a un nodo già esistente è proporzionale al grado di quest'ultimo. Collegamenti di questo tipo si dicono preferenziali e fanno si che si creino pochi nodi, chiamati *hub*, con un grado medio alto di circa 2 m, e molti nodi chiamati *periferici*, con grado medio compreso tra m e 2 m. In figura 2.7 è riportato un esempio.

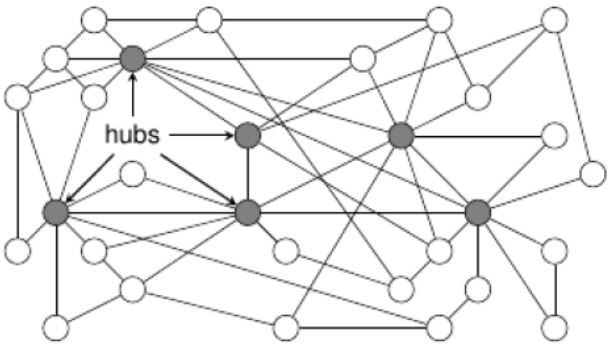


Figura 2.7: Esempio di Scale-free graph.

2.4 ALGORITMI DI GOSSIP

I protocolli Epidemici, o di Gossip, sono paradigmi computazionali e di comunicazione orientati a sistemi distribuiti su lar-

ga scala caratterizzati da alta dinamicità. Il metodo più semplice per la diffusione d'informazioni in una rete è effettuare un broadcast cercando di propagare al maggior numero di nodi l'informazione, ma con lo svantaggio di saturare tutti i canali di comunicazione. Per questo motivo i protocolli di gossip utilizzano un approccio probabilistico per la gestione della diffusione dell'informazione attraverso la rete, con lo scopo di massimizzare la diffusione dell'informazione sovraccaricando il meno possibile i canali di comunicazione. In un protocollo epidemico, un nodo che ha un'informazione sceglierà in modo casuale un nodo vicino con cui comunicare e tentare di condividere l'informazione.

Uno dei protocolli epidemici più famoso è "Game of life" [22], un automa cellulare sviluppato dal matematico inglese John Conway sul finire degli anni sessanta. L'ambiente per quest'algoritmo è un insieme di celle, dove ogni cella ha 8 celle vicine come mostrato in figura 2.5. Lo scopo dell'algoritmo è di diffondere il seme della vita, quindi ogni cella può essere occupata o no in base alle seguenti regole:

- Nascita: se una cella non occupata ha tre celle vicine occupate, diventa anch'essa occupata.
- Morte: se una cella occupata ha 0 o 1 cella vicina occupata, muore di solitudine, oppure se ha dalle 4 alle 8 celle vicine occupate, muore di sovraffollamento.
- Sopravvissuta: se una cella occupata ha 2 o 3 celle vicine occupate, essa sopravvive alla generazione successiva.

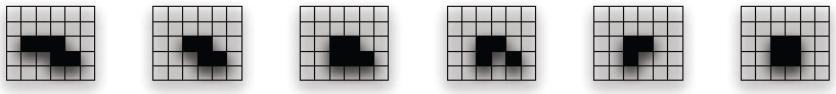


Figura 2.8: Algoritmo Game of Life

2.4.1 *Classificazione degli algoritmi epidemici*

Come discusso in [27] in ogni algoritmo epidemico vi è quella che è chiamata popolazione, un insieme di unità interattive comunicanti. Queste unità usano un set di regole che specificano le modalità di diffusione di una specifica informazione ad altre

unità. La scelta di queste regole è molto legata al tipo di algoritmo e al comportamento che quest'ultimo deve avere. L'importante è che a ogni istante di tempo t, ogni unità sia in uno dei seguenti tre stati, riguardo a una specifica informazione:

- Suscettible (suscettibile): l'unità non conosce nulla dell'informazione in questione ma è disposta a venirne a conoscenza.
- Infected (contagiata): l'unità è a completa conoscenza dell'informazione in questione e utilizza il set di regole per diffondere a sua volta l'informazione.
- Removed (Rimossa): l'unità è a completa conoscenza dell'informazione in questione ma non la diffonde.

Sulla base delle definizioni appena descritte, possiamo definire diverse classi di algoritmi, in cui è indicato per ogni classe come in generale sono trattate le informazioni.

Le principali classi sono:

- Suscettible – Infected (SI).
- Suscettible – Infected – Suscettible (SIS).
- Suscettible – Infected – Removed (SIR).

Esistono anche altre classi che estendono ulteriormente queste classi, aggiungendo alcuni stati intermedi.

In modelli ***Suscettible – Infected (SI)*** si ha che i nodi possono essere suscettibili a un'informazione e quando ne vengono a conoscenza, diventano contagiati e vi rimangono fintanto che tutta la popolazione non diventa contagiata. Ciò però richiede ulteriori controlli esterni per decidere quando terminare la diffusione dell'informazione.

A differenza del modello SI, in quello ***Suscettible – Infected – Suscettible (SIS)*** un'unità contagiata può decidere di fermare la diffusione dell'informazione prima che tutta la popolazione sia contagiata. Ogni unità rimossa può tornare a essere contagiata se riceve nuovamente l'informazione che aveva smesso di trasmettere e ricominciare a trasmetterla di nuovo finché non perde nuovamente interesse nel farlo.

Il modello ***Suscettible – Infected – Removed (SIR)*** è molto simile al modello SIS, con la differenza che un'unità rimossa rimane rimossa per sempre per quella determinata informazione e non potrà più esser contagiata da quell'informazione. Ciò non impedisce che tale unità possa diventare poi suscettibile a nuove informazioni.

2.4.2 *Medoti di diffusione*

In generale, ogni algoritmo di gossip prevede che a ogni iterazione dell'algoritmo ogni unità della popolazione se deve comunicare con un nodo, esso sia scelto in maniera casuale tra i nodi dell'intera popolazione. Poi ogni algoritmo specifica, con vincoli più stringenti, regole diverse di selezione del destinatario. In generale però vi sono tre metodi con cui le unità di una popolazione possono diffondere le informazioni ad altre unità:

- Push.
- Pull.
- Push&Pull.

Metodo Push
L'algoritmo Push prevede che i nodi contagiati prendano l'iniziativa di diffondere l'informazione, quindi a ogni istante t, il nodo contagiato sceglie un nodo casuale e prova a contagiarlo, come mostrato in figura 2.9. Questa strategia è molto efficace all'inizio della diffusione, quando vi è un alto numero di unità suscettibili e poche contagiate o rimosse, quindi la probabilità che ogni nodo contagiato ha di scegliere un nodo suscettibile è alta. Questa probabilità decresce col passare del tempo perché la quantità di nodi suscettibili diminuisce e il numero di nodi contagiati o rimossi aumenta, rendendo questo metodo poco affidabile nel lungo periodo. Questo metodo non garantisce che tutta la popolazione sia contagiata.

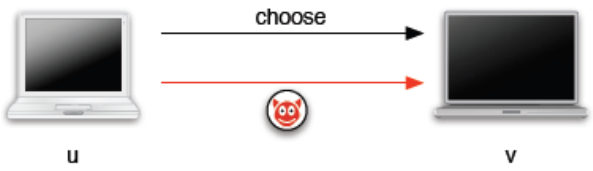


Figura 2.9: Metodo di Push.

Metodo Pull
L'algoritmo Pull prevede che un nodo contagiato non si muova attivamente nel diffondere l'informazione, ma che siano i nodi suscettibili a fare richiesta di nuove informazioni ai nodi contagiati. A ogni istante t, un nodo suscettibile seleziona casualmente un altro nodo e gli chiede se ha nuove informazioni.

Se il nodo contattato ne ha, allora restituisce l'informazione; in figura 2.10 è riportato un esempio. Quest'algoritmo di propagazione è poco efficace all'inizio dell'epidemia perché vi è solo un nodo contagiato e la probabilità di scegliere proprio lui è uno sulla grandezza della popolazione. Col passare del tempo, più l'informazione si diffonde, più alta sarà la probabilità di selezionare un nodo contagiato. Questo metodo non garantisce che il processo di diffusione abbia inizio poiché vi è una probabilità che nessun nodo suscettibile contatti il nodo contagiato, ma vi è anche la possibilità che tutti i nodi scelgano il nodo contagiato generando così un rapido inizio di epidemia. Questo metodo però garantisce che, se l'epidemia inizia, essa si diffonderà a tutta la popolazione.

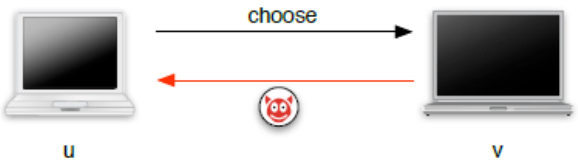


Figura 2.10: Metodo di Pull.

Metodo Push&Pull

I metodi Push e Pull presentano vantaggi e svantaggi in differenti momenti del processo di diffusione, l'unione dei due ha lo scopo di combinare i vantaggi dei due metodi. In questo caso un nodo contagiato utilizzerà una strategia Push, mentre i nodi suscettibili utilizzeranno una approccio Pull finché non saranno contagiati. Nelle prime fasi della diffusione premierà più la strategià Pulsh, mentre a contagio avanzato si sentirà di più la strategia di Pull. In figura 2.11 è rappresentato un esempio di strategia Push&Pull.

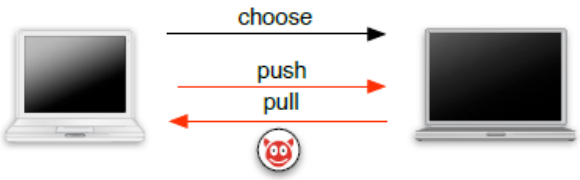


Figura 2.11: Metodo di Push&Pull.

2.4.3 *Applicazioni*

Il letteratura vi sono molti esempi di applicazioni degli algoritmi di gossip, tra i tanti citiamo quello per il corretto Replicated Database Maintenance [18], l'Epidemic Routing e il Mantenimento della visione del Network Overlay [23]. In merito al Replicated Database Maintenance il primo algoritmo studiato fu il Direct Mail, una soluzione che prevede che il nodo in possesso della nuova informazione debba contattare direttamente ogni altro nodo per diffondere l'aggiornamento. Questo metodo è posso affidabile perché è molto sensibile allo stato della rete su un unico nodo e sull'unico mittente. Per ovviare a tali problemi sono stati introdotti due principali algoritmi di gossip: Anti-Entropy e il Rumor Mongering. L'Anti-Entropy è un algoritmo di tipo SI, che cerca come dice il nome di contrastare la crescita dell'entropia in un sistema di replicazione dati, facendo si che a ogni ciclo dell'algoritmo, a coppie, i nodi sincronizzino i loro dati. Unico problema è che per la sincronizzazione di database, a volte è necessario trasmettere sulla rete l'intero database. Il Rumor Mongering invece è un algoritmo di tipo SIR che ha lo scopo di propagare tra i nodi solo l'elenco degli aggiornamenti e mai l'intero database. A ogni ciclo ogni nodo che ha la lista aggiornamento non vuota, sceglie un nodo casuale e cerca di propagargli i suoi aggiornamenti. Con questo tipo di algoritmo nasce il problema di trovare un criterio col quale decidere quando smettere di propagare gli aggiornamenti.

Da questo sono nati diverse versioni dell'algoritmo di Rumor Mongering, in base ai diversi criteri usati per fermare la propagazione degli aggiornamenti. E' quindi possibile suddividere la classe degli algoritmi di Rumor Mongering nelle seguenti sotto classi:

- ***Blind***: un nodo contagiato decide di interrompere la diffusione in base al suo stato interno.
- ***Feedbackind***: un nodo contattato risponde dicendo se già conosce oppure no l'informazione. Il nodo contagiato mittente, in base alla risposta prende una decisione.
- ***Coin***: il nodo contagiato si fermerà con una probabilità $1/k$, dove k è il numero di nodi contattati.
- ***Counter***: il nodo contagiato incrementa un contatore per ogni informazione inviata con successo. Si ferma quando il contatore supera una soglia prefissata.

2.4.4 *Algoritmi di gossip per il Peer2Peer*

Nel contesto degli algoritmi di gossip relativi a reti P2P, presenteremo quelli più utilizzati per la diffusione di messaggi e discussi in [26] e [12]. Essi sono il Probabilistic Broadcast (PB), Probabilistic Edge (PE) e Fixed Fanout (FF). Definiti i seguenti termini:

- Λ_i: insieme di nodi adiacente all'i-esimo nodo.
- V_i: la cardinalità di Λ_i.
- *msg*: il messaggio da diffondere.
- p_v,p_e,*fanout*: sono i valori probabilistici a ciascun algoritmo, rispettivamente PB,PE e FF.

Probabilistic Broadcast (PB)
Il Probabilistic Broadcast, prevedere che chi ha l'informazione o la ricevuta, la diffonda con una certa probabilità pv ai nodi vicini. L'algoritmo 1 descrive in pseudo codice il funzionamento di questo metodo.

Algorithm 1 Probabilistic Broadcast

```
1: function GossipPB(msg,p_v)
2:     if Random() ⩽ p_v then
3:         for all s_i ∈ Λ_i do
4:             Send(msg,s_j)
5:         end for
6:     end if
7: end function
```

Probabilistic Edge (PE)
Il Probabilistic Edge, prevedere che chi ha l'informazione o l'ha ricevuta, la diffonda su ogni arco uscente con una certa probabilità pe per ogni arco. L'algoritmo 2 è lo pseudo codice che descrive il funzionamento.

Fixed Fanout (FF)
Il Fixed Fanout, prevedere che chi ha l'informazione o l'ha ricevuta, la diffonda a fanout nodi adiacenti selezionati a caso tra tutti i nodi vicini. L'algoritmo 3 descrive tramite pseudo codice il funzionamento di questo metodo.

Algorithm 2 Probabilistic Edge

```
1: function GossipPE(msg,p_e)
2:     for all s_j ∈ Λ_i do
3:         if Random() ⩽ p_e then
4:             Send(msg,s_j)
5:         end if
6:     end for
7: end function
```

Algorithm 3 Fixed Fanout

```
1: function GossipPE(msg,fanout)
2:     if fanout ⩾ V_i then
3:         toSend ← Λ_i
4:     else
5:         toSend ← ∅
6:         for f = 1 to fanout do
7:             random select s_j ∈ Λ_i/toSend
8:             toSend ← toSend ⋃ s_j
9:         end for
10:    end if
11:    for all s_j ∈ toSend do
12:        Send(msg,s_j)
13:    end for
14: end function
```

2.5 SIMULATORI

La nostra ricerca si è focalizzata su simulatori di reti Peer-2-Peer, reti in cui tutti i nodi della rete sono allo stesso livello, sono paritetici e non vi è presenza di gerarchie tra nodi, e su simulatori di protocolli basati sia su cicli sia su eventi. In generale simulare qualcosa prevede l'esecuzione di una sequenza di operazioni nel tempo e, dove le entità che eseguono tali operazioni sono molteplici. Per poter gestire l'esecuzione contemporanea e/o concorrente di più entità il simulatore deve stabilire l'ordine in cui eseguire quale istruzione di quale nodo. La scelta sul come venga stabilito l'ordine di esecuzione caratterizza il motore del simulatore che potrà essere basato su cicli, su eventi o su entrambi. Una simulazione basata su cicli prevede che si definisca uno "time step" di simulazione, che è l'incremento del tempo virtuale di simulazione che definisce la cadenza di ogni ciclo. Tutte le operazioni definite come pe-

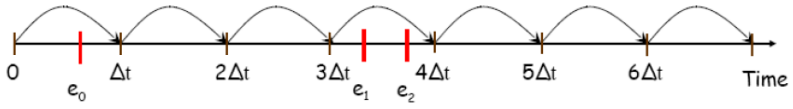


Figura 2.12: Scheduling guidato dai cicli.

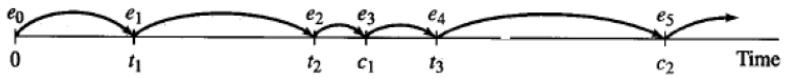


Figura 2.13: Scheduling guidato dagli eventi.

riodiche verranno eseguite ad ogni ciclo, contemporaneamente. Inoltre qualsiasi evento non periodico che occorre tra un ciclo e un altro verrà schedulato al ciclo successivo. Come si vede in figura 2.12, gli eventi e_0, e_1 ed e_2 accadono tra un ciclo e il successivo, con un passo di simulazione di Δt. L'evento eo verrà quindi processato all'instante Δt, mentre gli eventi e_1 ed e_2 verranno processati all'instante $4\Delta t$. In questo modo si perde la distribuzione degli eventi, avendo lo scheduler che appiattisce la distribuzione temporale su un unico ciclo, ad ogni iterazione. Questo tipo di simulazioni sono molto vincolanti e non permettono una rappresentazione realistica di una generica classe di nodi. Si adatta molto bene a simulazioni di operazioni batch e/o periodiche. Per poter avere più precisione e dettaglio nella modellazione dell'esecuzione degli eventi vi è lo scheduling guidato dagli eventi che, come dice il nome, ordinerà le operazioni da eseguire in base al momento in cui esse hanno generato o genereranno un evento.

In figura 2.13 è riportato un esempio di scheduling guidato dagli eventi. Il tempo di simulazione non sarà più una progressione costante, ma seguirà il susseguirsi degli eventi, nell'ordine in cui essi avvengono. In questo caso la simulazione avrà inizio con l'evento e_0, poi il simulatore salterà all'instante di tempo t1 in quanto lo scheduler prima di t_1 non ha nessuna operazione che deve essere eseguita.

All'istante t_1 avviene l'evento e_1 e viene processato, poi si passerà all'istante di tempo t_2 perché l'evento e_2 è previsto che avvenga in quell'istante e così via tutti gli altri. In generale gli eventi sono relativi per esempio alla ricezione di un messaggio da parte di un nodo. In figura 2.13 vediamo che vi sono gli eventi e_3 ed e_5 che sono schedulati agli istanti di tempo c_1 e c_2. Questo è un esempio di integrazione di eventi ciclici in un ambiente guidato dagli eventi. Gli eventi e_3 ed e_5 sono

azioni periodiche e quindi devono essere eseguite ogni "time step" specificato, ma come fare se non vi è un "time step" di simulazione? Il simulatore genererà un evento ogni "time step" di tempo di simulazione associando come evento l'azione periodica. Ciò garantisce di poter avere con un unico motore di scheduling, sia eventi aperiodici che azioni periodiche. In questo caso abbiamo un simulatore basato sia sugli eventi sia sui cicli.

In letteratura esistono tanti simulatori di reti come Mosquite[1] che è una libreria per CSIM, PADS[2] sviluppato dal Dipartimento di Ingegneria Informatica dell'Università di Bologna in grado di simulare protocolli complessi su reti di larga scala o senza una ben definita struttura. Vi è anche PeerSim[3] una libreria Java che fornisce gli strumenti di simulazione basata sia su cicli sia su eventi. Abbiamo poi concentato la nostra attenzione su OMNeT++[4], un framework costruito sulla piattaforma Eclipse[5] ma scritto in C++. OMNeT++ è uno simulatore basato solo su eventi ma dato il suo intenso sviluppo in ambito commerciale, vi sono disponibili diverse librerie e tool in grado di offrire molte funzionalità di simulazione di protocolli di rete e diverse tipologie di reti. Nel seguito presenteremo brevemente le caratteristiche di PeerSime e OMNeT++.

2.5.1 *PeerSim*

PeerSim è un software sviluppato in ambiente accademico che consente di simulare reti con una alta flessibilità sul numero di nodi che compongono la rete, rendendolo quindi adatto a simulare reti di grandi dimensioni. In letteratura troviamo diversi riferimenti e trattazioni in merito a questo software, come [21], [1] e [20]. PeerSim poi necessita che venga definito lo stack di protocolli che ogni nodo della rete implementerà; infatti la logica con cui questo simulatore costruisce la rete da simulare è quella di istanziare solo un nodo e poi esso viene copiato, o clonato, tante volte fino a raggiungere il numero di nodi nella rete. Per questo motivo si nota subito che PeerSim è uno strumento ottimizzato per la simulazione di reti anche di grandi dimensioni e soprattutto, come si può dedurre dal

1 http://www.mesquite.com/
2 http://pads.cs.unibo.it/doku.php?id=start/
3 http://peersim.sourceforge.net/
4 http://omnetpp.org/
5 https://eclipse.org/

nome, reti soprattutto Peer-to-Peer [20]. Non presenta alcun tipo di strumento grafico per la rappresentazione della rete o dei pacchetti in transito tra i nodi, la simulazione è solo computazione e stampa messaggi su console. Vi è la possibilità di utilizzare o estendere alcune funzioni di alcuni componenti del sistema di simulazione che permettono di stampare su file di testo le coordinate dei nodi della rete e/o i rami del grafo della rete nel formato usato da GNUplot[6] e quindi poi visualizzare il layout di rete. PeerSim necessita che l'utente definisca in un apposito file di configurazione le specifiche di simulazione, della rete, dei protocolli implementati sui nodi, delle dipendenze tra i protocolli. Richiede inoltre di specificare i componenti di inizializzazione, i componenti di controllo e i componenti di osservazione. I componenti di inizializzazione sono componenti che il simulatore eseguirà solo all'inizio del processo di simulazione e hanno il compito di inizializzare i nodi della rete e la rete stessa, ad esempio per stabilire le connessioni tra i nodi. I componenti di controllo sono "agenti" che vengono eseguiti ciclicamente e hanno il compito di agire sulla rete durante la simulazione, per introdurre dinamicità nella stessa come per esempio accendere o spegnere nodi o canali di trasmissione per simulare malfunzionamenti oppure aggiungere o togliere nodi e collegamenti. Questi componenti di controllo sono opzionali. Infine i componenti di osservazione detti anche osservatori sono componenti che operano alla fine della simulazione oppure ciclicamente se resi cycle-based impostando il loro time step. Il loro compito è quello appunto di osservare la rete e permettere di raccogliere dati per analisi. I dati da raccogliere devono essere specificati dall'utente nell'apposita funzione di osservazione che viene eseguita dal motore di simulazione. Un esempio di osservatore è il componente che permette di scrivere su file il layout della rete in formato compatibile con GNUplot. Punto forte di questo simulatore quindi sono la flessibilità sul numero di nodi della rete, le ottime prestazioni di simulazione anche per grandi reti e la scelta di lasciare all'utente l'estensione dei componenti che governano la simulazione.

2.5.2 *OMNeT++*

OMNeT++ è un framework e libreria di simulazione basata sul linguaggio C++, estendibile e modulare. OMNeT++ è un soft-

6 http://www.gnuplot.info/

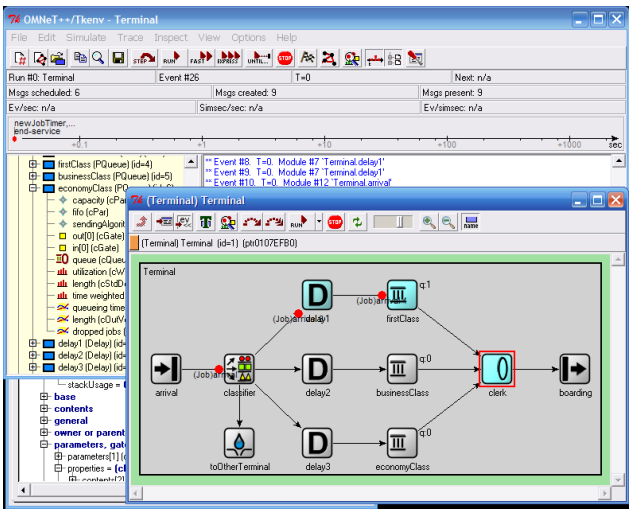


Figura 2.14: Esempio di rappresentazione tramite Tkenv.

ware di simulazione molto diffuso sia nel settore commerciale sia nel settore scientifico per la simulazione di reti e protocolli di trasmissione, disponibile dal 1997. OMNeT++ è un software che offre un editor di sviluppo basato su Eclipse. Sono disponibili una gran varietà di strutture base come reti wired e wireless ed è possibile aggiungere estensioni che permettono di ampliare la gamma di reti supportate. INET è una delle estensioni più corpose del framework e contiene una grossa quantità di reti e protocolli delle più diffuse strutture di rete utilizzate. Vi sono molte trattazioni in letteratura riguardo a questo strumento, come [3], [5] fino a uno studio per uso didattico [4].

OMNeT++ è un simulatore di eventi discreti, ma come spiegato all'inizio di questa sezione, è possibile coprire anche situazioni di esecuzione di azioni periodiche, organizzandole come eventi programmati. OMNeT++ contiene anche un ambiente grafico [5], Tkenv, per la rappresentazione grafica della struttura della rete e degli eventuali link fisici tra i nodi; è in grado inoltre di eseguire un ri-ordinamento spaziale dei nodi della rete con l'obiettivo di rappresentare il grafico della rete col minor numero di intersezioni tra i collegamenti possibili, per rendere più leggibili lo schema ed eventuali animazioni date dalla simulazione. In figura 2.14 è riportato un esempio. OMNeT++ offre una architettura modulare, cercando di perseguire il principio di riutilizzo dei moduli[5]. Ogni modulo corrisponde ad un singolo componente della rete o a un componente aggregato, composto a suoa volta da componenti singoli. In questo modo è possibile costruire reti su più livelli, e poter analizzare la rete da punti di vista di diverse granularità. Il linguaggio di alto

livello utilizzato per descrivere i moduli si chiama Network Definition (NED). La struttura è definita in linguaggio NED, mentre il comportamento dei moduli e dei componenti è scritto utilizzando sempre il C++, nei rispettivi file. Con OMNeT++, come per PeerSim, è possibile specificare in appositi file di inizializzazione una lista di parametri necessari a configurare la rete durante la sua fase di inizializzazione prima di iniziare la simulazione vera e propria. In un singolo file di inizializzazione è possibile specificare più scenari di esecuzione e per ognuno specificare quindi valori diversi. I parametri specificati possono essere letti dal file di Network Definition e usati per creare e impostare la rete secondo i parametri specificati dall'utente nel file di inizializzazione. Tramite il file di Network Definition si può propagare la lettura dei parametri fino ai file C++ con le procedure che modellano il comportamento del componente. OMNeT++ offre anche un sistema di raccolta dati, tramite il quale si possono poi analizzare i dati raccolti, filtrarli e ottenere differenti rappresentazioni grafiche dei dati stessi [5]. La raccolta dati può essere sia la sequenza temporale dei risultati, sia una raccolta delle statistiche su dati specifici oppure entrambe le cose.

2.5.2.1 *File di Inizializzazione*

Un file di inizializzazione è un file necessario alla corretta configurazione della rete, dei suoi componenti e dell'ambiente di simulazione. Un file di inizializzazione ha estensione ".ini" e per comodità verrà anche chiamato "INI file". In un file di inizializzazione è possibile fornire parametri organizzati in una struttura ad albero, in cui ogni ramo è uno scenario di simulazione indipendente dagli altri ma eredita tutti i parametri coi rispettivi valori, se non vengono ridefiniti, dello scenario padre. Al momento della simulazione vera e propria, il sistema chiederà all'utente di scegliere quale scenario di simulazione utilizzare se non ve ne è uno specificato direttamente nello script che lancia l'esecuzione di simulazione. Nella figura 2.15 sono riportati due esempi di file d'inizializzazione. Vediamo nella figura 2.15a come sotto la sezione "General" vengano specificati tutti i parametri comuni a tutte le configurazioni e di seguito vengono specificate tutte le altre configurazioni coi parametri specifici per quelle simulazioni. Mentre nella figura 2.15b è riportata una possibile struttura ad albero delle configurazioni.

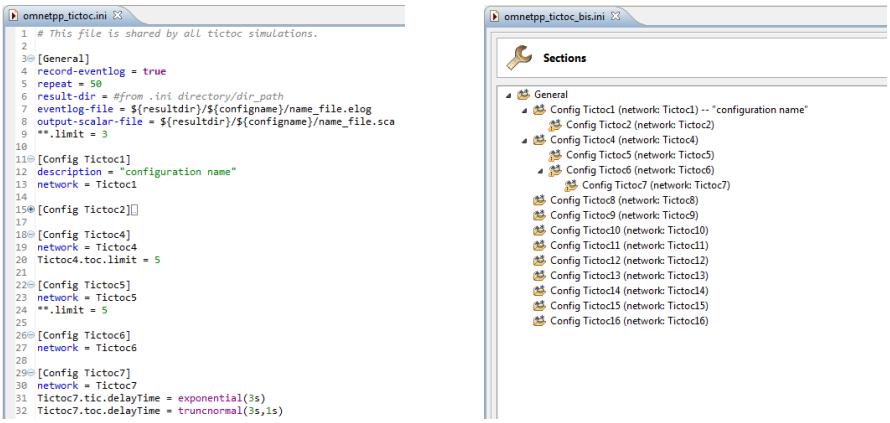
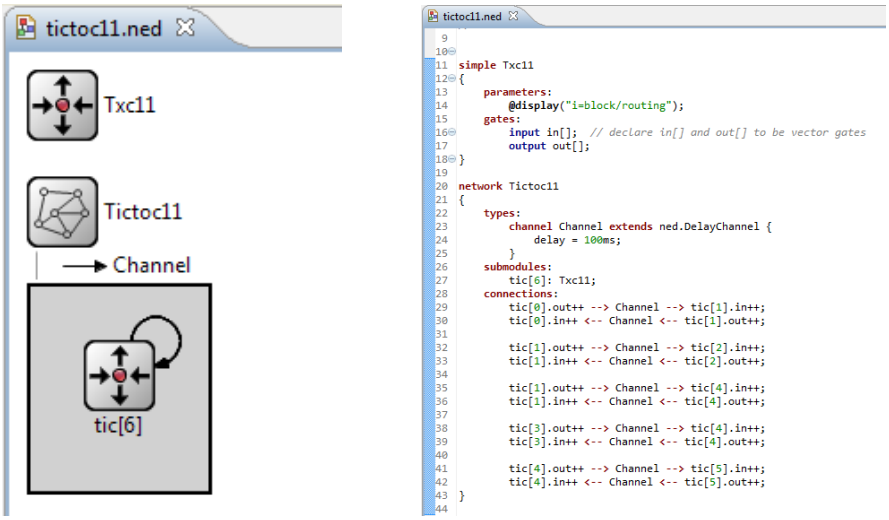


Figura 2.15: Esempio di file di inizializzazione.

In generale ogni file di inizializzazione caratterizza un set di scenari da simulare, quindi si tende a includere nella sezione "General" anche tutti i parametri che servono a impostare correttamente la simulazione o la gestione del framework quali, per esempio l'unità di misura con la quale viene scandito il tempo di simulazione, il numero di ripetizioni per ogni caso di studio che si desidera fare, le varie directory in cui si vogliono salvare i risultati o i dati raccolti.

2.5.2.2 *Linguaggio NED*

OMNeT++ offre, tramite un linguaggio ad alto livello chiamato Network Definition, la possibilità di definire il layout della rete e la struttura dei singoli componenti e della composizione dei componenti più grandi. Il linguaggio di Network Definition viene usato per creare file di specifica con estensione ".ned", che abbrevieremo con NED file. Il linguaggio NED è stato pensato per poter implementare singoli moduli in maniera indipendente dall'ambiente, rendendo così facile il loro riutilizzo o estensione. Il sistema ci offre due visibilità del layout di rete ad alto livello: Design e Source. Con la vista Design viene fornita una rappresentazione grafica della rete, se pur sempre ad alto livello, mentre la vista Source permette di vedere e implementare il file come puro codice sorgente. In figura 2.16a è riportato un esempio di NED file in vista Design, dove il sistema dice che nel file in questione sono stati usati componenti di tipo "Txc11", i canali di comunicazioni utilizzati sono dei "Channel" e la rete si può riassumere come un vettore di nodi collegati tra di essi tramite diversi "Channel". La vera rappresentazione del layout



(a) Vista Design. **(b)** Vista Source.

Figura 2.16: Esempio di file NED.

di rete viene fatta solo a runtime in fase di simulazione nel caso venga utilizzato lo strumento grafico Tkenv. In figura figura 2.16b invece è riportato lo stesso file della figura 2.16a ma in vista Source. Come si può vedere nella figura figura 2.16b, in vista sorgente possiamo vedere tutto il codice corrispondente alla vista Design. Si nota che nella prima parte viene definito il singolo componente "Txc11", poi viene definita la rete come un vettore di componenti "Txc11" e tra essi vi sono dei collegamenti di tipo "Channel", che è definito dentro la rete.

2.5.2.3 *Raccolta Dati*

OMNeT++ fornisce metodi per la raccolta di dati e statistiche e la possibilità di analizzarli al termine della simulazione. Si possono raccogliere due categorie di dati: i log temporali, chiamati anche event-log e i dati relativi a statistiche. I log temporali non sono altro che rappresentazioni grafiche della successione degli eventi tra i nodi della rete. Molto utile per analizzare la sequenza temporale degli eventi, la loro durata e le loro interazioni, soprattutto quando si hanno parecchi eventi che avvengono in contemporanea; in figura 2.17 è riportato uno frammento di un event-log. Per quanto riguarda le statistiche, si possono raccogliere dati con riferimento temporale, chiamati vettori, oppure statistiche su dati, chiamati scalari, quali media, varianza, deviazione standard, somma, minimo, massimo e altri ancora. Sia

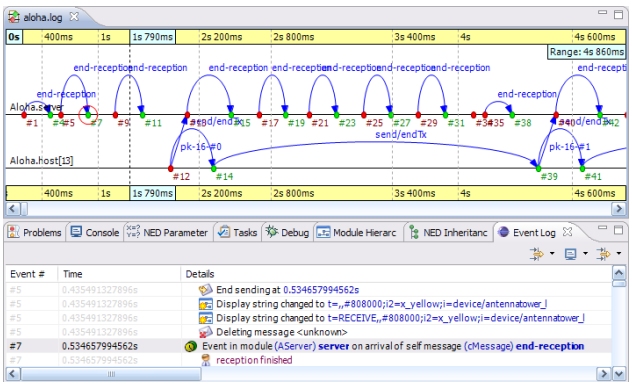


Figura 2.17: Esempio di event-log.

sui vettori che sugli scalari è possibile poi fare delle operazioni di manipolazione dati, come eseguire raggruppamenti, applicare filtri, applicare operazioni ai dati o a gruppi di dati come l'operatore media. Si può applicare l'operatore media a un grafico temporale di vettori, per ottenere l'andamento medio, utile per visualizzare il comportamento asintotico del sistema; in figura 2.18 è riportato un esempio. In figura 2.19 è riportato un esempio di rappresentazione grafica di vettori temporali più la rappresentazione dell'andamento medio. In figura 2.20 invece è riportato un esempio di istogramma come rappresentazione della distribuzione del conteggio di hop necessari per raggiungere hostess destinatario.

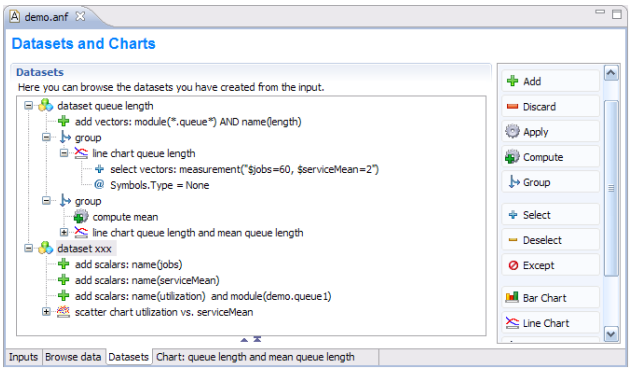


Figura 2.18: Esempio di manipolazione dati.

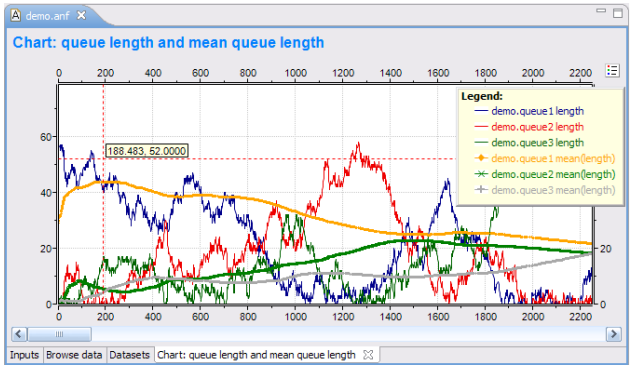


Figura 2.19: Esempio di grafico temporale di vettori e medie.

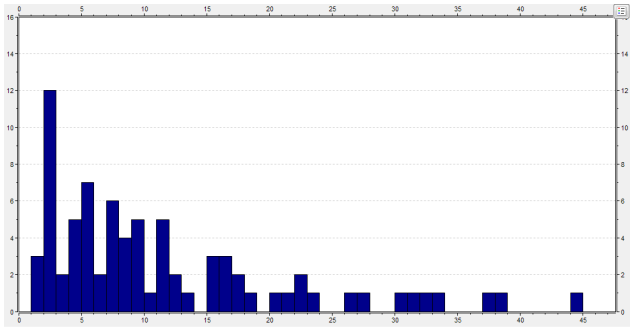


Figura 2.20: Esempio di istogramma, grafico per dati scalari.

3

IMPOSTAZIONE DEL PROBLEMA DI RICERCA

4

PROGETTAZIONE LOGICA DELLA SOLUZIONE PROPOSTA

Il nostro studio era mirato alla ricerca un possibile modo di diffondere messaggi e informazioni tra la popolazione, in situazioni dove le principali reti di comunicazione non sono disponibili. Ci siamo quindi concentrati su ciò che rimane disponibile in questi scenari e quali siano i metodi per modellare la situazione. Ci siamo concentrati sullo sfruttare ciò che è di più comune uso, ciò che è molto diffuso e facilmente in possesso di ogni persona. Da questo studio è risultato che gli smartphone rispecchiano questo profilo, infatti, al giorno d'oggi in media ogni persona possiede uno smartphone. Siamo passati quindi a studiare ciò che uno smartphone potesse fornire o ciò che di utile avremmo potuto sfruttare. Uno smartphone, ma in generale ogni dispositivo mobile, senza la presenza delle comuni reti di comunicazione, quali internet e telefonia, ha solo altri due modi di poter comunicare con altri dispositivi mobili: la tecnologia a infrarossi e la tecnologia Bluetooth. Da un rapido studio è emerso che la scelta degli infrarossi non avrebbe avuto portato a una soluzione performante e applicabile, mentre il Bluetooth si è rivelato essere un ottimo punto di partenza per un caso di studio. Il sistema Bluetooth in generale, come presentato nella Sezione 2.2, è un sistema di comunicazione Peer-to-Peer e quindi permetterebbe di mettere in comunicazione due dispositivi mobili anche in assenza delle reti di comunicazione. A questo punto ci siamo concentrati sul capire come poter sfruttare questa tecnologia per progettare una soluzione. Gli aspetti principali studiati sono: il modello di rete da utilizzare, il metodo con cui guidare e gestire la diffusione dell'informazione e infine come tutto ciò vada a impattare sul consumo energetico dei dispositivi. Un altro aspetto fondamentale di questo lavoro è stato tenere in considerazione il consumo energetico aggiuntivo che si va a introdurre su singolo dispositivo, poiché se l'impatto energetico fosse troppo elevato, si avrebbe una considerevole riduzione nell'autonomia del dispositivo stesso, rendendo la soluzione proposta impraticabile.

Abbiamo fatto, come prima cosa, uno studio sul consumo energetico richiesto da una singola trasmissione dati tramite

BLE. Abbiamo poi studiato diversi casi, in cui abbiamo fatto variare la grandezza del pacchetto trasmesso e valutato i consumi energetici richiesti. Dal nostro studio è emerso che l'energia richiesta per il trasferimento di una singola informazione, anche di discrete dimensioni, non vada a inficiare in maniera gravosa il consumo energetico medio dei più comuni smartphone in commercio. Lo standard BLE utilizza molti parametri per qualsiasi evento di trasmissione o ricezione. Abbiamo utilizzato Excel per costruire un modello in grado di simulare calcolare il tempo impiegato per la trasmissione di un'informazione, specificandone la grandezza e tutti i parametri che caratterizzano la trasmissione BLE. Molti parametri possono variare su intervalli piuttosto larghi, ma come illustrato nella documentazione ufficiale [8] e anche in [11], più i valori dei parametri crescono più il consumo energetico richiesto mediamente diminuisce. Questo perché aumentano i tempi di riposo e diminuiscono i momenti in cui il dispositivo deve rendersi attivo per trasmettere o ricevere; ovviamente questo va a discapito delle prestazioni in termini di ritardi. Per questo motivo, nel nostro studio abbiamo mantenuto tutti i parametri ai loro valori minimi per studiare il caso di massimo consumo energetico medio e anche di massima prestazione.

Nel nostro studio abbiamo fatto variare la grandezza dell'informazione da pochi Byte fino a 200 MB, in modo da vedere il comportamento con informazioni grandi e realmente possibili. In figura 4.1 riportiamo su grafico i risultati ottenuti e, come si può notare, dopo un certo valore, l'andamento del consumo energetico rimane lineare e diretto con la grandezza dell'informazione. L'andamento iniziale molto basso è dovuto alla predominazione dei tempi di attesa e connessione dello standard BLE sul tempo di trasmissione dell'informazione. La trasmissione dati su BLE prevede che dopo un'attesa variabile iniziale, la trasmissione avvenga tramite connection event consecutivi; un connection event è un evento nel quale sono scambiati dati tra due dispositivi connessi tra loro. Infine abbiamo stimato quante trasmissioni un dispositivo possa affrontare, oltre al suo carico medio di lavoro prima di scaricarsi. Il calcolo prevede che si parta sempre a batteria completamente carica e abbiamo considerato i più comuni e attuali smartphone sul mercato. La stima ovviamente dipende dalla grandezza dell'informazione e varia da un minimo attorno alle 200 trasmissioni fino a un massimo nell'intorno delle 400 trasmissioni.

Lo studio sul consumo energetico ci ha confermato che il bas-

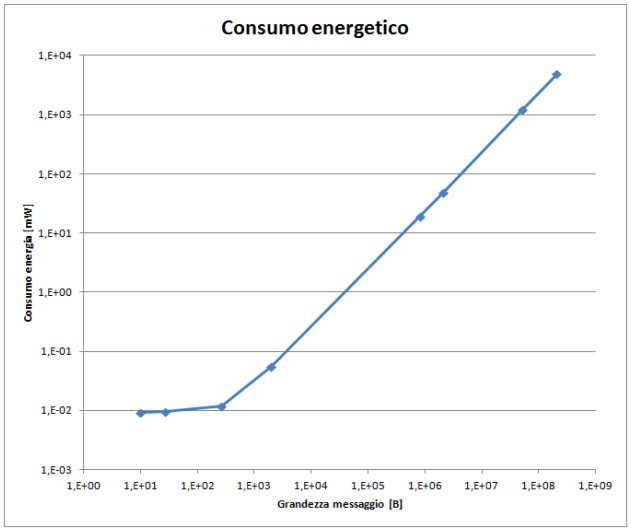


Figura 4.1: Energia media richiesta per il trasferimento di una informazione.

so livello di energia richiesto dalla tecnologia BLE per la trasmissione di informazioni, anche di medie dimensioni, permette di sfruttare questo sistema di comunicazione per diffondere messaggi tra nodi mobili. Dato il poco consumo energetico, la nostra soluzione prevede che il dispositivo mobile non debba dedicarsi completamente a questo servizio, anzi esso rimarrà sempre a disposizione dell'utente senza che il nostro sistema limiti all'utente l'uso del dispositivo. Questo perché ormai sui dispositivi mobile, quali smartphone o tablet ad esempio, vi sono tante altre applicazioni o funzioni utili all'utente che debbano rimanere fruibili in ogni situazione, anche in scenari come quelli da noi ipotizzati.

Dopo l'aspetto energetico, ci siamo dedicati allo studio di un modello di rete che potesse descrivere correttamente la disposizione dei dispositivi e del mezzo di comunicazione. Il BLE, ma la tecnologia Bluetooth in generale, è un sistema di comunicazione molto legato alla potenza del trasmettitore e dalla relativa distanza che tale potenza consente di raggiungere. Per questo motivo abbiamo scelto di modellare la nostra rete di dispositivi mobili con una Random Geometric Graph, presentata nella Sezione 2.3.2. La scelta di questo modello di rete è stata abbastanza naturale, visto la comune caratteristica che caratterizza la trasmissione BT e il modello di rete stesso.

Infine dopo l'aspetto del modello di rete, ci siamo dedicati allo studio di un sistema di regole che potessero governare la diffusione delle informazioni. Dopo un'attenta analisi è emer-

so che per com'è composta la rete e per il tipo d'informazioni, un modo efficace di diffondere informazioni è quello del passa parola. Per questo motivo abbiamo preso in considerazione il mondo degli algoritmi di gossip, o epidemici, creati per modellare la diffusione d'informazioni tra le persone e sui social network, o nel caso degli epidemici come si diffonde un'epidemia in una popolazione. Tra gli algoritmi di gossip, abbiamo scelto di estendere l'algoritmo del Fixed Fanout, un algoritmo di gossip di tipo Rumor Mongering presentato nella Sezione 2.4.4. La nostra soluzione è stata progettata perché tenga in considerazione il consumo energetico che essa genera sul dispositivo. Per questo motivo abbiamo progettato il nostro algoritmo, estendendo quello già esistente, aggiungendo elementi molto dinamici. Infatti, i parametri da noi progettati, regolano l'algoritmo in modo dinamico, permettendogli di adattarsi allo stato dell'ambiente esterno e allo stato interno del dispositivo. Per stato dell'ambiente esterno intendiamo il numero di dispositivi presenti nelle vicinanze, quindi lo stato della rete attorno al dispositivo, mentre per stato interno intendiamo il livello della batteria, in modo da regolare lo "sforzo" in maniera corretta. Il nostro algoritmo cerca sempre di regolare i parametri in modo da garantire un buon compromesso tra prestazioni e consumo energetico/carico di lavoro. Dopo un'attenta analisi abbiamo deciso di progettare l'algoritmo con due differenti valori di reattività, di fronte a cambiamenti nella rete o nello stato interno del dispositivo oppure in entrambi.

4.1 MODELLO DI RETE

Dopo aver ricercato in letteratura, abbiamo capito che il modello di rete di tipo Peer-to-Peer è adatto a modellare la rete che abbiamo studiato. Le P2P sono reti di nodi totalmente paritetici, senza alcuna struttura gerarchica o differenziazione tra i nodi stessi. Per questo motivo ogni nodo della rete è chiamato peer e lo scambio d'informazioni avviene sempre tra due peer alla volta. Tra i tanti modelli presenti in letteratura, abbiamo scelto quello che riesce a rappresentare al meglio le caratteristiche della rete da noi in esame: il modello che abbiamo scelto è il Random Geometrig Graph (RGG). Questo tipo di modello è costituito da un grafo $G(N,\rho)$ bidirezionale casuale inserito in un'area limitata ed è composto da N nodi. Il parametro ρ rappresenta la distanza massima entro la quale sono stabiliti i

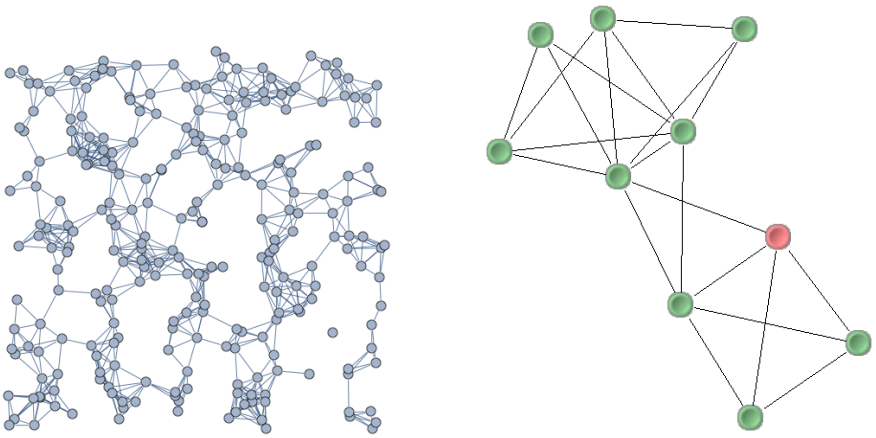


Figura 4.2: Esempio di file di inizializzazione.

collegamenti tra i nodi. Il raggio ρ è di fatto la distanza geometrica entro la quale un nodo stabilisce collegamenti con altri nodi. Due nodi a una distanza inferiore di ρ avranno un collegamento bidirezionale, mentre due nodi a una distanza maggiore di ρ non avranno nessun collegamento tra loro. Questo tipo di modello presenta una basa degree variance, ma un'alta edge dependency. Sono due parametri che rispettivamente indicano la varianza nella distribuzione del grado dei nodi e un fattore di dipendenza dagli archi del grafo che indica la probabilità che vi siano collegamenti tra nodi vicini.

Nel nostro caso di studio il raggio ρ modella la portata del trasmettitore Bluetooth, infatti, solo dispositivi che si trovano entro il raggio d'azione dei rispettivi trasmettitori possono comunicare, mentre dispositivi fuori portata non avranno alcun collegamento. La costruzione di un RGG inizia scegliendo le caratteristiche dell'area, base e altezza. Successivamente si dispongono in maniera casuale uniforme i nodi del grafo all'interno dell'area. Infine si controllano le distanze tra tutti i nodi e si instaurano i collegamenti solo trai i nodi a una distanza uguale o inferiore a ρ. In figura **??** sono riportati due esempi di RGG.

Com'è possibile intuire, il parametro ρ è la chiave principale che determina quanto sia connesso il grafo. Più ρ è piccolo più la probabilità di avere sottografi isolati e/o archi *bridge* tra i sottografi è elevata. Un bridge è un arco singolo che collega un nodo di un sottografo con un nodo di un altro sottografo, rendendo quell'arco l'unica possibile via di comunicazione tra i due sottografi. Un raggio ridotto, diminuisce la possibilità di

avere collegamenti ridondanti per la diffusione dell'informazione e aumenta la possibilità di avere gruppi di nodi isolati e la formazione di bottleneck nella rete. Al contrario, più il raggio cresce più la probabilità di avere nodi o sottogruppi di nodi isolati diminuisce, come anche la probabilità di avere archi bottleneck e la rete tenderà a essere sempre più connessa. Va detto anche che oltre al raggio ρ, è altrettanto importante la dimensione dell'area in cui la rete si trova. Nonostante l'area non sia un parametro del grafo, essa influisce, insieme a ρ, a rendere il grafo più o meno connesso. Il motivo è dovuto al metodo con cui è generato il grafo. Presa un'area, i nodi vengono distribuiti in modo casuale e uniforme all'interno dell'area. Più l'area è grande, più la distribuzione uniforme tenderà a disperdere i nodi. Dato un grafo con N e ρ fissati, più l'area è grande più la densità dei nodi per metro quadro sarà bassa, mentre al contrario, più l'area è piccola più la densità sarà elevata. Questo fattore di dispersione ci ha permesso di poter simulare scenari diversi, dal grosso centro urbano con alta densità abitativa, fino al paese di campagna dove la popolazione è molto più diradata. Per questo motivo, è stato importante lavorare secondo ipotesi di diverse densità di nodi; fissati densità e numero di nodi, l'area è calcolata di conseguenza. Come già detto in precedenza, il nostro algoritmo presenta una componente dinamica, che tiene in considerazione le variazioni dell'ambiente esterno. Negli scenari da noi considerati, è molto facile che la densità di popolazione possa variare e per questo motivo abbiamo implementato moduli specifici che hanno lo scopo di seguire questi cambiamenti.

4.2 SISTEMA DI TRASMISSIONE

Abbiamo scelto di utilizzare il Bluetooth Low Energy come mezzo di comunicazione perché, in situazioni di assenza di tutte le altre reti di comunicazione, esso rimane comunque operativo. Il BLE, presentato nella Sezione 2.2, è uno standard di comunicazione Peer-To-Peer ufficializzato nel 2010. Nella sua versione 4.0, la casa produttrice ha presentato uno standard denominato Low Energy, infatti, questa nuova tecnologia ha consumi ridotti anche di dieci volte rispetto alle sue versioni precedenti. Ciò ha contribuito alla sempre maggior diffusione e utilizzo di questo standard nei settori del monitoraggio ambientale, della sensoristica, dei dispositivi wearable in coppia con i dispositivi mobile. Questa tecnologia è stata sviluppata per lo

scambio di piccoli dati tra due enti e quindi ottimizzando i consumi energetici per questo tipo di carico di lavoro; il BLE non è stato progettato per lo streaming. Questo standard presenta ridotti tempi di latenza e parametri di connessione più semplici, rendendo più veloce connettersi a un dispositivo. Questo standard è equipaggiato su ogni dispositivo mobile in commercio e i modelli più recenti sono già dotati della versione 4.1. Da un indagine ISTAT [17] del 18 Dicembre 2014, le percentuali sulla presenza della tecnologia all'interno delle famiglie italiane sono tutte in crescita, come la presenza di una connessione a Internet salita 64% o di una connessione a banda larga salita a quasi il 63%, ma dato più significativo per noi è stato quello dei telefoni cellulari, che sono presenti in più del 93% delle famiglie italiane. L'uso di Internet tramite smartphone è cresciuto dal 20.8% al 28% e ciò sta a indicare un rinnovamento anche nei dispositivi stessi; le persone acquistano dispositivi più recenti che permettano loro di navigare più facilmente su Internet. Sempre dall'analisi ISTAT è emerso che il 22.4% delle persone che navigano su Internet dai 14 anni in su lo ha fatto tramite un computer, mentre il 35.4% degli utenti tramite cellulare o smartphone e solo il 6% da altri dispositivi mobili. L'utilizzo dello smartphone è la tipologia di dispositivi più utilizzata per l'accesso a Internet. Per questo motivo abbiamo formulato l'ipotesi che ogni persona possieda uno smartphone e che possa essere rappresentata da esso nella nostra rete. Per questi motivi abbiamo scelto come strumento di comunicazione la tecnologia BLE.

Richiamiamo brevemente la macchina a stati che rappresenta il funzionamento del Link Layer del BLE. Essa è composta dai seguenti cinque stati:

- **Standby**: stato raggiungibile da tutti gli altri stati. In questo stato non è possibile effettuare nessuna trasmissione di pacchetti né riceverne.
- **Advertising**: un dispositivo in questo stato si chiama advertiser e può inviare pacchetti di advertiser. Resta in ascolto per pacchetti in risposta ai suoi pacchetti di advertisinge.
- **Scanning**: un dispositivo in questo stato si chiama scannar e ricompre il ruolo di osservatore. Rimane in ascolto per pacchetti di advertising.

- **Initiating**: stato raggiungibile da tutti gli altri stati. In questo stato non è possibile effettuare nessuna trasmissione di pacchetti né riceverne.
- **Standby**: un dispositivo in questo stato si chiama initiator e rimane in ascolto di pacchetti di advertising. Risponderà a quei pacchetti di advertising ai quali è interessato, con un pacchetto di *connection request* con l'intenzione di aprire una connessione.
- **Connection**: lo stato di Connection può essere raggiunto sia dallo stato di Advertising sia dallo stato di Initiating. Indica che il dispositivo sta tentando di connettersi o è connesso con un altro dispositivo. Lo stato Connection si suddivide in due ruoli: Slave e Master.
  - **Slave**: il ruolo di Slave è affidato a chi arriva nello stato di Connection dallo stato di Advertising. Utilizza i parametri di connessione decisi/inviategli dal Master.
  - **Master**: il ruolo di Master è affidato a chi arriva nello stato di Connection dallo stato di Initiator. Un dispositivo Master decide i parametri della connessione, che invierà allo Slave.

Nell'implementazione della nostra soluzione abbiamo mappato uno a uno gli stati del BLE con gli stati dell'algoritmo di gossip scelto, che descriveremo nella sezione successiva, in modo da avere una corrispondenza tra stato di trasmissione con stato di contagio. In figura 4.3 richiamiamo la macchina a stati del BLE.

4.3 SOLUZIONE: ALGORITMO DYNAMIC FANOUT

Dopo una profonda analisi degli algoritmi di gossip presenti in letteratura e della loro classificazione, abbiamo deciso di basare la nostra soluzione su algoritmi di tipo Rumor Mongering, in altre parole algoritmi che hanno come obiettivo la diffusione di un "rumore" o pettegolezzo, nel nostro caso un'informazione. Nella Sezione 2.4.4 abbiamo presentato alcuni algoritmi di gossip adatti a reti P2P e i diversi aspetti che li caratterizzano. La nostra scelta è stata di usare come base l'algoritmo del Fixed Fanout, per poi estenderlo con l'aggiunta di parametri dinamici, allo scopo di dare appunto dinamicità all'algoritmo e capacità

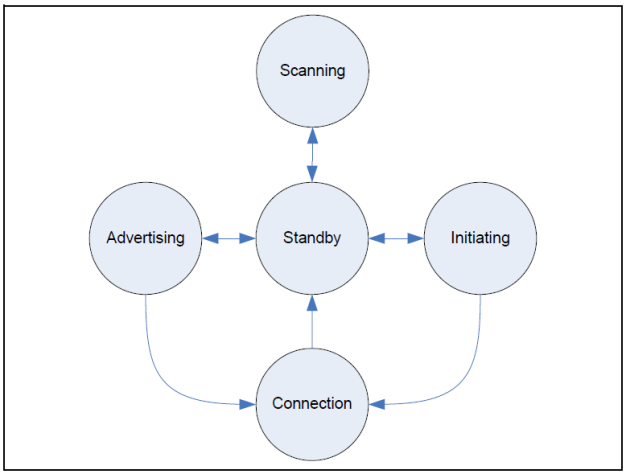


Figura 4.3: Macchina a stati del Link Layer del BLE.

di adattamento ai diversi cambiamenti dell'ambiente esterno e dello stato interno dei dispositivi stessi. Con cambiamenti dell'ambiente estero intendiamo le variazioni della rete nel raggio d'azione del nodo, quindi il variare del numero di nodi che esso percepisce, mentre con variazioni interne al dispositivo intendiamo la carica/scarica della batteria. Questo perché abbiamo cercato di progettare i parametri in modo che abbiano sempre un valore che possa garantire prestazioni accettabili e che calibri il carico di lavoro del dispositivo in relazione al livello energetico che la batteria ha. Se il livello di energia è elevato, l'algoritmo cerca di assegnare al dispositivo un carico di lavoro maggiore se necessario, mentre se il livello di energia è medio basso, l'algoritmo comincia a calibrare i parametri in modo da trovare un compromesso di carico di lavoro ed efficienza in modo da non gravare molto sulla restante poca autonomia del dispositivo. Dato che il sistema non è né un sistema dedicato né occupa completamente le risorse di calcolo del dispositivo, non vogliamo privare l'utente degli eventuali altri servizi presenti sul suo smartphone che possano servirgli. Per questo motivo l'algoritmo, dopo che la batteria scende sotto una certa soglia limite, pone il sistema in stato di Standby per non intaccare l'autonomia residua del dispositivo.

Brevemente richiamiamo le caratteristiche dell'algoritmo di Fixed Fanout. E' un algoritmo di gossip di Rumor Mongering di tipo Suscettible – Infected – Removed (SIR) , con criterio di terminazione di tipo "counter". Questo significa che nel momento di inviare una nuova informazione, l'algoritmo sceglie un nodo casuale dall'insieme dei suoi vicini e tenta di trasferirgli l'informazione. Se il trasferimento va a buon fine, incremen-

ta un contatore di uno. Ripete quest'operazione fino a quanto il conteggio raggiunge un valore limite chiamato *fanout*. Il sistema esegue esattamente *fanout* trasferimenti e poi passa in stato di Rimosso. Non vi è nessun elemento di controllo su cambiamenti della rete, né vi sono elementi in grado di variare il valore di *fanout*. La nostra estensione riguarda principalmente due parametri che hanno lo scopo di sostituire le componenti statiche dell'algoritmo originale con componenti dinamiche. La nostra soluzione rimane un algoritmo di tipo SIR, ma con l'aggiunta di controlli finalizzati alla riduzione dello spreco energetico. Nell'implementazione del nostro algoritmo, abbiamo associato a ogni stato dell'automa del Link Layer del BLE uno stato del paradigma SIR. Il mapping è il seguente:

- Suscettibile $\longleftrightarrow$ Initiating.
- Contagiato $\longleftrightarrow$ Advertising.
- Rimosso $\longleftrightarrow$ Standby.

Questa è l'associazione teorica generale, relativa alla generica informazione. All'atto pratico, si aggiungono allo stato di Rimosso anche gli stati di Initiating e di Scanning, perché quando un dispositivo perde interesse a diffondere un'informazione, esso può anche entrare in stato di Initiating alla ricerca o in attesa di nuove informazioni da condividere, oppure passare in stato di Scanning se vuole solo ascoltare la rete passivamente.

I parametri che abbiamo progettato rappresentato due diversi criteri di terminazione: uno di tipo counter e uno di tipo blind. In questo modo il sistema ha controlli sia esterni che interni. I parametri in questione sono:

- Dynamic Fanout.
- Advertising Limit.

Il Dynamic Fanout ha la stessa funzione del precedente *fanout*, rappresentare il limite di trasmissioni che il dispositivo può compiere. Nel nostro algoritmo però, il Dynamic Fanout viene calcolato dinamicamente e tenuto aggiornato in maniera periodica.

L'Advertising Limit è un nuovo parametro di terminazione, introdotto e progettato allo scopo di far capire al dispositivo, quale sia la situazione della rete attorno a lui in termini di contagio. Serve a segnalare al nodo quando i nodi attorno a lui non sono più interessati alla sua informazione. Anche l'Advertising Limit è aggiornato periodicamente.

4.3.1 *Dyanmic Fanout (DF)*

Il Dyanmic Fanout (DF) ha la stessa funzione del precedente *fanout*, rappresentare il limite di trasmissioni che il dispositivo può compiere. Nel nostro algoritmo però, il DF è calcolato dinamicamente e tenuto aggiornato in maniera periodica. Nell'algoritmo di FF, il dispositivo continua a tentare a trasmettere l'informazione finché raggiunge il limite fissato di *fanout*. Nella nostra soluzione invece, il dispositivo prova a trasmettere fino a DF volte, ma non vi è obbligato perché vi sono altri controlli in grado di stabilire quando fermare la diffusione perché il sistema ha rilevato inutile continuare. Questo controllo coinvolge l'Advertising Limit. Il DF è composto di due fattori, uno che tiene conto del livello della batteria (parte Blind) e uno che tiene conto dello stato della rete attorno al dispositivo quindi il numero di nodi percepiti (limite per il Counter). Il componente che tiene conto della batteria rappresenta un fattore di partizionamento che sarà usato per scegliere il limite di trasmissioni a seconda di quanti nodi il dispositivo percepisce. Ad esempio, se il sistema ha la batteria totalmente carica, il fattore di partizionamento sarà del 50%, mentre se la batteria è meno della metà, sarà ad esempio del 20%. Nel capitolo successivo, esporremo più nel dettaglio come questo parametro è calcolato e come si caratterizza l'andamento dinamico.

4.3.2 *Advertising Limit (AL)*

L'Advertising Limit (AL) è un nuovo parametro di terminazione, introdotto e progettato allo scopo di far capire al dispositivo, quale sia la situazione della rete attorno a lui in termini di contagi. Questo parametro rappresenta il limite di pacchetti di advertising che possono andare a vuoto consecutivamente. Quando un nodo riceve una nuova informazione da condividere, esso inizia le trasmissioni e comincia a inviare l'informazione ad altri nodi, uno alla volta. Al termine di ogni trasmissione, se il limite DF non è ancora stato raggiunto, il nodo inizierà a inviare nuovamente pacchetti di advertising alla ricerca di altri nodi ancora da contagiare. Il sistema, tramite l'uso di timeout, capisce quando un pacchetto di advertising va a vuoto e, nel caso, ne invia un altro. Allo stesso tempo conteggia il numero di pacchetti di advertising che vanno a vuoto consecutivamente. Se questo conteggio raggiunte l'AL, il nodo si ferma e passa in stato di Rimosso o di Initiating per le motivazioni sopra dette,

esattamente come se avesse effettuato tutte le DF transazioni. Lo scopo principale di questo parametro è di intervenire la dove il controllo DF è cieco, cioè nel controllore se tra i dispositivi che il sistema percepisce, c'è qualcuno realmente interessato all'informazione. Potrebbe accadere che la rete cambi proprio dopo aver aggiornato i parametri e quindi un dispositivo si ritroverebbe con un DF grande e non coerente con il reale stato della rete, fino al prossimo aggiornamento. In queste situazioni l'AL entra in gioco e impedisce al dispositivo di sprecare batteria, qualora l'AL sia raggiunto. Come il DF anche l'AL è composto di un fattore per il livello della batteria e di un fattore riguardante lo stato della rete circostante il nodo. Come sarà poi spiegato nel capitolo successivo, il fattore riguardante la batteria è stato tenuto neutro, poiché l'energia richiesta per fare advertising è molto esigua e quindi abbiamo deciso di usare un fattore di partizionamento neutro; la funzione inserita per il fattore relativo ai nodi percepiti è sufficientemente restrittiva.

4.4 SIMULAZIONE

Durante la fase di studio e progettazione iniziale, abbiamo analizzato diversi sistemi e piattaforme di simulazione in grado di simulare reti come quella da noi studiata. Si faccia riferimento alla Sezione 2.5 per la presentazione generale. La nostra scelta è ricaduta su OMNeT++, un framework e libreria di simulazione basata sul linguaggio C++, estendibile e modulare. OMNeT++ è un software di simulazione molto diffuso sia nel settore commerciale sia nel settore scientifico per la simulazione di reti e protocolli di trasmissione. OMNeT++ è un software che offre un editor di sviluppo basato su Eclipse Sono disponibili una gran varietà di strutture base come reti wired e wireless ed è possibile aggiungere estensioni che permettono di ampliare la gamma di reti supportate. INET è una delle estensioni più corpose del framework e contiene una grossa quantità di reti e protocolli delle più diffuse strutture di rete utilizzate.

Lo sviluppo anche in ambito commerciale ha reso questo strumento molto ricco di funzionalità e strumenti e anche di componenti di simulazione. OMNeT++ è fornito di un motore di simulazione event based ma come spiegato anche nella Sezione 2.5, è in grado di gestire anche azioni cicliche. Grazie al suo tool grafico Tkenv, è possibile visualizzare con animazioni la rete e i pacchetti in movimento in essa; vi è anche la possibilità di modificare alcuni aspetti visivi su Tkenv, tramite appo-

site istruzioni, per avere a runtime maggior espressività e potenzialità della simulazione stessa. OMNeT++ permette anche un'ottima gestione della parte di layout della rete tramite il suo linguaggio di alto livello chiamato Network Definition (NED), che permette di esprimere la struttura della rete in modo semplice e permette anche l'utilizzo di funzioni di casualità, così da poter creare una rete casuale. Il generatore di numeri casuali, per permettere di poter ripetere una simulazione nelle stesse condizioni, utilizza sempre gli stessi seed. Ciò significa che la disposizione causale che si ottiene eseguendo la simulazione, sarà uguale alla disposizione ottenuta eseguendo nuovamente la simulazione da capo una volta terminata la prima. Per ottenere diverse strutture causali è necessario dire al simulatore di fare più simulazioni data la configurazione. OMNeT++ poi, necessità l'implementazione di almeno un file di configurazione nel quale specificare tutte le configurazioni di simulazione e i valori dei parametri che la simulazione richiede. Anche in questo caso è possibile utilizzare funzioni casuali messe a disposizione dal sistema per assegnare valori ai parametri. Infine OMNeT++ ha già un sistema interno per la raccolta dati e l'analisi di statistiche fatte sui dati raccolti. Permette anche di visualizzare i dati raccolti e le eventuali statistiche su grafici. Inoltre permette anche di eseguire manipolazioni sui dati raccolti, come la possibilità di raggruppare dati, eseguire operazioni di media o varianza e tante altre; è possibile inoltre per l'utente specificare particolari operazioni tramite l'apposita sintassi del simulatore. Nel Capitolo 5 sono spiegati, più in dettaglio, gli aspetti riguardanti l'utilizzo dei file di Network Definition, dei file di configurazione e delle loro caratteristiche.

5

ARCHITETTURA DEL SISTEMA

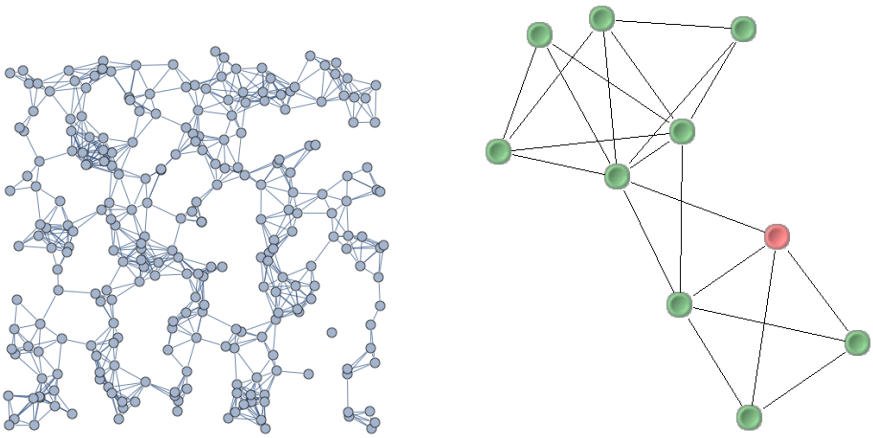
5.1 MODELLO DI RETE

Il modello di rete utilizzato è quello del Random Geometrig Graph (RGG), una rete di nodi rappresentata da un grafo connesso. I nodi sono connessi tra loro utilizzando il criterio di distanza geometrica, quindi ogni nodo sarà collegato con tutti quei nodi che sono ad una distanza uguale o inferiore di un dato raggio ρ da essi. Il raggio ρ sarà una variabile del nostro problema. Nella figura 5.1 sono rappresentati due esempi di RGG; la figura 5.1b è un esempio di rete RGG rappresentata dal tool grafico Tkenv di OMNeT++.

La generazione della nostra rete dipende da:

- Densità dei nodi.
- Numero di nodi che compongono la rete.
- Raggio d'azione del BLE (ρ).

La scelta di operare a diversi livelli di densità è dovuta al fatto di aver voluto studiare il comportamento della nostra soluzione in alcuni scenari urbani possibili per diverse distribuzioni geografiche sul territorio, in altre parole poter studiare il variare del comportamento del nostro algoritmo sia in situazioni di centri urbani normalmente molto densamente popolati, ad esempio medie/grandi città, sia in centri urbani con una più bassa concentrazione urbana che rappresentano la maggioranza dei paesi italiani. Per queste considerazioni, abbiamo ipotizzato di associare al singolo nodo un abitante, perché mediamente ogni persona possiede uno smartphone, come riportato anche dai dati presentati dall'ISTAT [17]. Abbiamo scelto come prima e più grande densità 0.02 abitanti per metro quadro perché ci è sembrato una densità già sufficientemente alta per avere ottimi valori in termini di perfomance, quindi scegliere densità superiori non ci è sembrato utile per il nostro scopo. Quello che invece abbiamo voluto studiare è come si degradano le prestazioni al diradarsi della concentrazione urbana, ed ecco il perché della scelta delle alte densità.



(a) (b)

Figura 5.1: Esempio di file di inizializzazione.

Le densità scelte sono:

- d = 0.02 $\frac{\text{nodi}}{\text{m}^2}$.
- d = 0.01 $\frac{\text{nodi}}{\text{m}^2}$.
- d = 0.008 $\frac{\text{nodi}}{\text{m}^2}$.
- d = 0.001 $\frac{\text{nodi}}{\text{m}^2}$.
- d = 0.0005 $\frac{\text{nodi}}{\text{m}^2}$.
- d = 0.0001 $\frac{\text{nodi}}{\text{m}^2}$.

La generazione dell'area dipende anche dal numero di nodi in quanto, per mantenere un certo valore di densità è necessario che il rapporto tra il numero di utenti e l'area analizzata sia corretto. Quindi per ogni densità abbiamo scelto di fare molteplici prove con diversi valori di utenti. I valori di N che vanno da 2 a 100, sono valori pensati per simulare situazioni reali, mentre i valori più grandi sono stati scelti per vedere la scalabilità della nostra selezione in casi estremi.

I valori per il numero di nodi scelti sono:

- N = 2.
- N = 5.

- N = 10.
- N = 30.
- N = 50.
- N = 80.
- N = 100.
- N = 200.
- N = 500.
- N = 1000.

Come già anticipato, una volta fissata la densità e il numero di nodi si ricavano le misure dell'area in analisi. Abbiamo scelto di operare in questo modo perché scegliere direttamente la grandezza dell'area senza vincoli, avrebbe portato alla scelta di aree troppo piccole o troppo grandi. Un area troppo piccola per il numero di nodi ha come conseguenza di presentare prestazioni perfette sempre, mentre scegliere un'area troppo grande restituirebbe la non utilità del sistema, a causa della troppa dispersione spaziale dei nodi. Scenari di questo tipo non sono oggetto delle nostre analisi.

Infine l'ultimo parametro che abbiamo voluto far variare è stato il raggio d'azione ρ del BLE. Le specifiche tecniche non danno nessuna indicazione della distanza coperta da questa tecnologia, perché, rispetto al precedente BT Classic, la scelta di questo fattore viene delegato alle aziende produttrici dei trasmettitori. La distanza dipende da quanto potente viene costruito un trasmettitore. Lo studio [2] ha stimato che in media la distanza è di 50 m, ma altri studi hanno rilevato che si può arrivare anche ad oltre 100 m in linea d'aria. Per il nostro caso di studi, poiché i dispositivi in analisi sono dispositivi mobili come smarthphone e tablet, abbiamo scelto di prendere un valore pessimistico e uno ottimistico. Questo perché, i trasmettitori equipaggiati sui dispositivi mobili non sono tutti uguali e quindi la scelta di studiare due valori agli estremi ci è parsa logica. I valori di ρ scelti sono:

- ρ = 15 m.
- ρ = 50 m.

Abbiamo scelto di utilizzare come valore più piccolo, un raggio poco più grande di quello della Classe 2 del Bluetooth Classic, che è di norma la distanza raggiunta dai dispositivi cellulari precedenti gli smartphone. Come termine massimo invece, abbiamo scelto di utilizzare il valore medio stimato dallo studio [2]. Ci è sembrato sensato che un normale smartphone non vada oltre i 50 m per un motivo: gli smartphone e i tablet sono soggetti ad una progettazione minimalista che cerca di compattare il più possibile in pochi millimetri di spessore. Questo comporta ad installare trasmettitori di potenza sufficiente per coprire l'utilizzo dei componenti accessori che il mercato offre (dispositivi wearable) e che possano occupare lo spazio desiderato dai designer, a differenza dei trasmettitori progettati per applicazioni in ambito industriale, i quali devono coprire distanze anche superiori a 100 m a volte, vengono progettati con l'obiettivo primario della distanza, senza vincoli di volume.

5.1.1 *Costruzione della rete*

In fase di simulazione, la generazione della rete viene fatta in modo automatico dal simulatore come la distribuzione dei nodi nell'area e la costruzione dei collegamenti tra i nodi. I vari parametri necessari, grandezza dell'area, numero dei nodi e raggio ρ sono inseriti negli appositi file d'inizializzazione. Quando viene eseguito lo scrip di lancio, viene caricato il file di inizializzazione corrispondente al caso di simulazione e nel file di Network Definition, che contiene i metodi per la costruzione della rete, vengono prima posizionati tutti i nodi della rete in maniera casuale e uniforme, poi nella sezione *connections* vi è il metodo che controlla e inserice i collegamenti tra i nodi. Dopo aver disposto casualmente i nodi nell'area, per ogni nodo controlla quali altri nodi si trovano ad una distanza geometrica inferiore al valore di ρ specificato nel file di inizializzazione e tra essi inserisce un canale di comunicazione con le prestazioni di trasmissione del BLE. *Nell'appendice x è riportato il codice del relativo file di Network Definition.*

5.1.2 *Algoritmo Dynamic Fanout*

Come presentato nella Sezione 4 la nostra soluzione è una estensione dell'algoritmo di gossip Fixed Fanout (fvedere Sezione **??**). Abbiamo esteso il metodo di calcolo per il limite del numero di trasmissioni che ogni nodo può effettuare, inserendo

nuove componenti dinamiche che hanno reso lo stesso fattore reattivo ai cambiamenti sia ambientali sia interni al dispositivo. Per questo motivo abbiamo chiamato il nostro algoritmo: Dynamic Fanout. Successivamente, abbiamo aggiunto un controllo per la terminazione delle trasmissioni che vengono segnalate inutili, permettendo all'algoritmo di limitare gli sprechi di energia. In questo modo il sistema cerca sempre di trovare un punto di lavoro che offra un compromesso tra efficienza e risparmio energetico nella maggior parte degli scenari studiati.

I due principali fattori che abbiamo progettato sono il Dynamic Fanout e l'Advertising Limit. Sono due parametri che rappresentano due criteri di terminazione di tipo contatore, ma che al loro interno hanno anche una componente di tipo "blind" necessaria a valutare lo stato della batteria del dispositivo. Il Dynamic Fanout rappresenta il limite di trasmissioni che il dispositivo può fare nel suo stato, mentre l'Advertising Limit rappresenta il limite di trasmissioni di advertising andate a vuoto consecutivamente che il dispositivo deve fare per decidere che nessuno dei suoi vicini è più interessato ad averla.

Il nostro algoritmo agisce basandosi sulla macchina a stati del BLE, cambiando stato all'avvenire di particolari eventi. Avendo fatto una relazione uno a uno tra gli stati dell'algoritmo di gossip e quelli della macchina a stati del BLE, abbiamo potuto centralizzare la gestione sul Bluetooth. Anche se il BLE per definizione è un sistema di trasmissione che consuma poca energia, abbiamo deciso che il nostro sistema sia in uno stato di attiva operatività finché la batteria del dispositivo è superiore al 10%. Questo perché, non vogliamo che il sistema vada a consumare le ultime riserve di energia del dispositivo, permettendo all'utente di usufruire di tutti quei servizi sul dispositivo di cui ha bisogno e di raggiungere eventualmente una fonte di ricarica. Inoltre la maggior parte degli smartphone implementa già sistemi di risparmio energetico che disabilitano sistemi non essenziali quando la batteria scende sotto certi livelli. Il nostro algoritmo, qualora la batteria scenda sotto il 10%, finisce l'eventuale trasmissione in corso e poi si porta in stato di Standby. Quando la batteria torna sopra il 10%, il sistema torna attivo e operativo. Questo controllo viene fatto periodicamente, tramite una procedura che esegue le "azioni periodiche". Sono azioni che devono essere eseguite periodicamente, indipendentemente dall'occorrenza di eventi esterni. Queste azioni periodiche prevedono il controllo dello stato della batteria e l'aggiornamento dei parametri. Questo processo rimane attivo anche

quando la batteria scende sotto la soglia limite e continua i suoi aggiornamenti, così da poter riattivare subito il sistema quando la batteria viene caricata. Nel nostro caso di studio abbiamo simulato la decadenza della batteria tramite il decremento di una variabile numerica. Un decremento costante ad ogni azione periodica per simulare il normale consumo della batteria, più un consumo ad ogni trasmissione/ricezione che il dispositivo effettua, per accentuare il fatto che il consumo energetico del sistema è concentrato nelle trasmissioni e non nel suo stato di attesa. Abbiamo scelto di decrementare la batteria ad ogni azione periodica, quindi col risultato di un consumo energetico medio più alto di quello reale, per studiare come si comporta il sistema con dispositivi soggetti ad un alto consumo energetico esterno al nostro sistema. Lo pseudo codice dell'algoritmo 4 descrive il funzionamento delle azioni periodiche, che vengono schedulate periodicamente dal sistema. La variabile booleana *busy* segnala quando il dispositivo è impegnato nell'esecuzione di qualche trasmissione o ricezione. All'Appendice A.1.1, riportiamo il diagramma di flusso della macchina a stati Standby. Questo diagramma sintetizza il comportamento del sistema nei momenti di inattività. All'avvenire di azioni periodiche, il sistema controlla il livello di batteria e se maggiore del 10%, porta il sistema in stato di Initiating. Se il sistema riceve dall'utente il comando di inviare un messaggio, esso passa in modalità di invio di un nuovo messaggio e l'algoritmo 5 descrive in pseudo codice le operazioni necessarie. All'Appendice A.1.2 riportiamo il diagramma di flusso della relativa macchina a stati.

Algorithm 4 Azioni Periodiche

```
1:  function AzioniPeriodiche(btState)
2:      decrementaBatteriaIdle()
3:      batteria ← livelloBatteria()
4:      if (busy || batteria ⩾ 0) then
5:          AggiornaParametri(batteria)
6:          if btState = STANDBY then
7:              btState ← INITIATING
8:          end if
9:      else
10:         btState ← STANDBY
11:     end if
12: end function
```

Algorithm 5 Invio Messaggio

```
1: function InvioMessaggio(btState,msg)
2:     if btState ≠ STANDBY then
3:         btState ← STANDBY
4:     end if
5:     Tx counter ← 0          ▷ Contatore trasmissioni effettuate.
   START:
6:     AE counter ← 0          ▷ Contatore degli Advertising Event.
7:     btState ← ADVERTISING
8:     AEcounter ← +1
9:     begin
10:        ADVERTISING EVENT
11:    end
12:    if èTimeoutScaduto() then
13:        if AE counter < AL then    ▷ AL = Advertising Limit.
14:            go to START
15:        else
16:            btState ← STANDBY
17:            AzioniPeriodiche(btState)
18:        end if
19:    end if
20:    busy ← VERO
21:    btState ← CONNECTION_SLAVE
22:    begin
23:        CONNECTION EVENT
24:    end
25:    Tx counter ← +1
26:    decrementaBatteriaTx()
27:    busy ← FALSO
28:    btState ← STANDBY
29:    if Tx counter < DF then
30:        go to START
31:    else
32:        AzioniPeriodiche(btState)
33:    end if
34: end function
```

Per inviare un messaggio, il sistema entra si porta in stato di Standby se non lo fosse già stato, inizializza le variabili contatore per Advertising Event (AE) e Trasmissionipoi e passa in stato di Advertising. A questo punto incrementa il contatore degli Advertising Event e inizia l'AE vero e proprio. Dopo aver trasmetto i suoi pacchetti, si mette in attesa di una risposta. Se il timeout scatta prima che una richiesta di connessione arrivi, allora quell'AE viene considerato a vuoto. Viene quindi controllato che il nuovo di AE a vuoto non sia superiore all'AL e poi si ricomincia con un altro AE. Se invece il AE counter è maggiore o uguale dell'AL, il sistema si ferma e si riporta sulle azioni periodiche e quindi in stato di ascolto (Initiating). Se invece, prima che scada il timeout, riceve una richiesta di connessione, allora il sistema si dichiara *busy*, occupato, si porta nello stato di Connection col ruolo di Slave e attende che il Connection Event cominci. Una volta terminata la trasmissione, incrementa il contatore delle trasmissioni, resetta il contatore degli Advertising Event, si dichiara non più occupato e si porta in stato di Standby. Infine il sistema controlla se ha effettuato DF transazioni. In caso negativo si riporta all'inizio di tutta la procedura, in caso positivo esce e riprende le azioni periodiche. Se il sistema rileva la presenza di un nuovo messaggio sulla rete, esso passa in modalità di ricezione di un nuovo messaggio e l'algoritmo 6 descrive in pseudo codice le operazioni necessarie. All'Appendice A.1.3 riportiamo il diagramma di flusso della relativa macchina a stati. Il fatto di rilevare un informazione nuova presuppone che il sistema sia in nello stato di Initiating. Se il sistema sta nello stato di Initiating, automaticamente resta in ascolto di nuove informazioni. Una volta nello stato di ascolto, quando rileva una nuova informazione, il sistema manda una richiesta di connessione al mittente del pacchetto di advertising. Dopo di che si segna come occupato ed entra nello stato di Connection col ruolo di Master e invia la richiesta di apertura del Connection Event. Se il timeout scade, vuol dire che la richiesta di connessione non è stata accettata e quindi il sistema si dichiara non più occupato e si riporta in stato di ascolto. Se invece la richiesta di connessione viene accetta, alla richiesta di apertura del Connection Event segue in risposta l'informazione, così da entrare definitivamente nel Connection Event. Completata la ricezione del messaggio, esso viene salvato, il sistema si dichiara non occupato e poi inizia ad inviare a sua volta questo nuovo messaggio.

Algorithm 6 Ricevi Messaggio

```
1:  function RiceviMessaggio(btState)
2:      if btState = INITIATING then
3:          go to SEND
4:      end if
    START:
5:      btState ← STANDBY
6:      btState ← INITIATING
    SEND:
7:      invioRichiestaConnessione()
8:      busy ← VERO
9:      btState ← CONNECTION_MASTER
10:     inviaRichiestaPool()          ▷ Richiesta d'apertura di C.E.
11:     if èTimeoutScaduto() then
12:         busy ← FALSO    ▷ La richiesta non è stata accettata.
13:         go to START
14:     end if
15:     begin
16:         msg ← CONNECTIONEVENT
17:     end
18:     decrementaBatteriaTx()
19:     busy ← FALSO
20:     InvioMessaggio (msg,btState)
21: end function
```

5.1.3 *Dyanmic Fanout (DF)*

Il Dynamic Fanout ha il compito di fermare il dispositivo dopo un certo numero di trasmissioni effettuate con successo per una certa informazione. Questo valore limite è calcolato in modo dinamico, dipendente dal livello di batteria del dispositivo e dal numero di nodi che il sistema riesce a percepire, quindi al numero di nodi ai quali può potenzialmente connettersi. Abbiamo voluto che questo parametro avesse un particolare andamento e rapidità di risposta in particolari situazioni e altri andamenti per altre situazioni. A tale scopo, abbiamo provato a modellare il comportamento di questo parametro attraverso diverse funzioni ottenendo quindi comportamenti più o meno conservativi, secondo la funzione scelta.

Il calcolo del DF è composto da due fattori: un primo fattore che tiene conto del livello di batteria del dispositivo, chiamato *Fattore Batteria* e da un secondo fattore dipendente dal numero

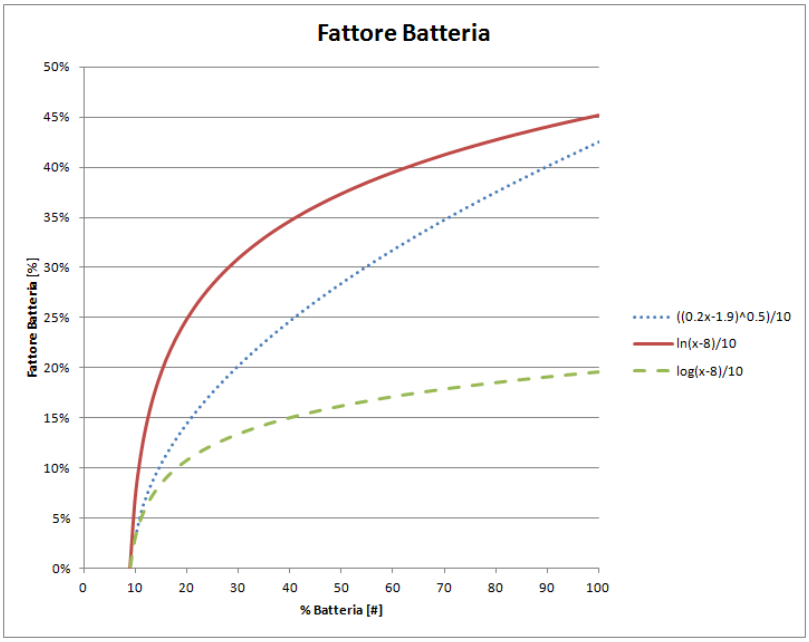


Figura 5.2: Curve delle Funzioni Batteria.

di nodi, che ha la funzione di correggere l'andamento globale della funzione risultante. La nostra idea è di ottenere dal Fattore Batteria una percentuale che rappresenta la quantità di nodi, tra quelli percepiti, che il dispositivo prenderà in considerazione come suo limite di trasmissione cioè come suo DF. Più batteria un dispositivo ha più alta sarà la percentuale di nodi cui potremo trasmettere l'informazione. In figura 5.2 riportiamo in grafico tre funzioni batteria studiate, ognuna con un andamento diverso; esse sono rispettivamente:

$$y = \frac{\sqrt{0,2 \cdot x - 1,9}}{10} \tag{5.1}$$

$$y = \frac{\ln(x - 8)}{10} \tag{5.2}$$

$$y = \frac{\log(x - 8)}{10} \tag{5.3}$$

Tutte e tre le funzioni sono divise per dieci, così da ottenere un valore percentuale. Dalla figura 5.2 si nota come diversi tipi di funzioni diano differenti curve di risposta, più o meno conservative e più o meno reattive per valori tra 10% e il 20% di batteria. Noi abbiamo scelto di utilizzare la funzione

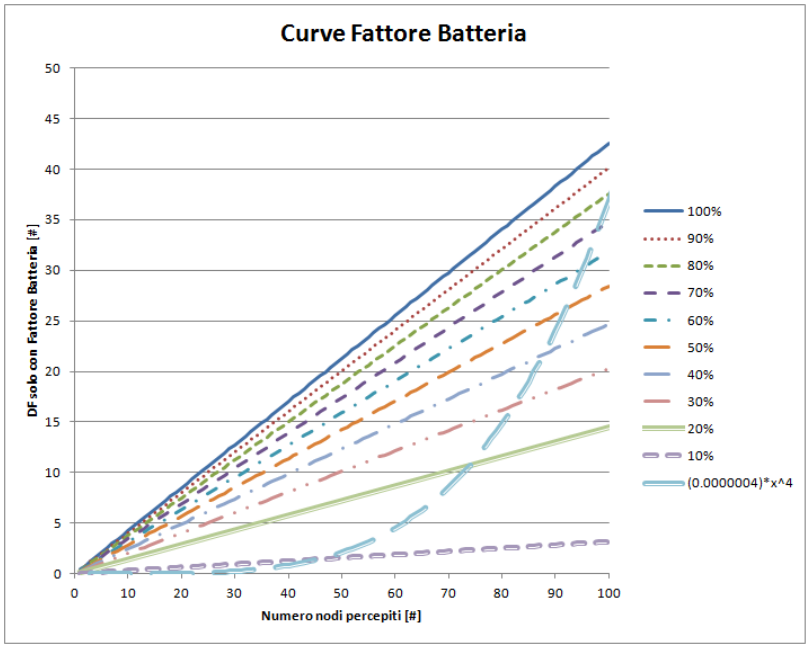


Figura 5.3: Applicazione del Fattore Batteria al numero di nodi e Fattore di Correzione.

di Equazione (5.1) rispetto alla funzione col logaritmo naturale, Equazione (5.2), poiché non troppo aggressiva per valori bassi di batteria ma nemmeno eccessivamente conservativa e perché al crescere della percentuale di batteria le due funzioni tendono allo stesso valore massimo o quasi. L'Equazione (5.3) risulta essere troppo conservativa e non scala bene con la crescita della rete, non permettendo una buona diffusione delle informazioni.

La funzione scelta per il Fattore Batteria è quindi:

$$FB = \frac{\sqrt{0,2 \cdot x - 1,9}}{10} \tag{5.4}$$

La variabile x rappresenta la percentuale di batteria del dispositivo ed è compresa nell'intervallo $[0;100] \in \mathbb{R}$.

Il Fattore Batteria quindi indica la percentuale sulla totalità dei nodi percepiti, da utilizzare come limite di trasmissioni. L'Equazione (5.4) restituisce un numero valore tra 0 e 1, che applicato al numero di nodi percepiti, genera una retta linearmente crescente, al crescere del numero dei dispositivi percepiti. In figura 5.3 abbiamo riportato le principali rette, una ogni 10% di batteria. Come si può notare ogni retta è monotona crescente e ciò non va bene, perché si avrebbero limiti di trasmissioni totalmente inadeguati al crescere della rete. Abbiamo quindi

pensato di introdurre un fattore di correzione, che aiutasse a bilanciare la monotona crescita di queste rette. Più il numero di nodi è alto, più questo fattore correttivo è forte.

Il fattore di correzione ha la seguente equazione:

$$FC = 0.0000004x^4 \tag{5.5}$$

La variabile x rappresenta il numero di nodi percepiti dal dispositivo e $x \in \mathbb{R}, x \geqslant 0$

Abbiamo progettato il Fattore di Correzione in modo che possa risultare neutro per valori bassi di numero di nodi, mentre applichi la sua funzione correttiva per valori medio-grandi e grandi di numero di nodi. Questo è il motivo per cui abbiamo scelto un coefficiente così piccolo. In questo modo al crescere del numero di nodi, partendo da un valori piccoli, la componente dominante è il Fattore Batteria, poi al crescere dei nodi, la componente dominante diventa sempre più quella correttiva. Il motivo per cui abbiamo scelto di inserire questo fattore di correzione, è che ci serviva qualcosa per gestire la monotona crescita dovuta al solo fattore batteria e anche per scegliere meglio, con più criterio, il valore del DF. Col crescere della rete, cresce anche la possibilità che vi siano più dispositivi nella rete con un alto livello di batteria e quindi è possibile spalmare meglio il carico di lavoro su molti più nodi. Con una rete meno densa, lasciamo che il DF sia più grande per coprire gli eventuali dispositivi vicini che posso avere poca batteria e quindi partecipare poco o addirittura nulla al processo di diffusione. Con una rete sempre più densa invece, possiamo ridurre questo "sovraccarico", perché anche avendo la possibilità di nodi vicini con poca batteria, cresce la probabilità che vi siano altri nodi con molta batteria che possono affrontare anch'essi un buon carico di lavoro. Per questo motivo, per un alto numero di nodi, abbiamo un DF decrescente. Unendo quindi, Fattore Batteria e Fattore correttivo abbiamo il seguente primo calcolo del DF.

$$\begin{aligned} DF &= 1 + FB \cdot x - FC \\ &= 1 + \left(\frac{\sqrt{0,2 \cdot z - 1,9}}{10} \right) \cdot x - 0.0000004x^4 \end{aligned} \tag{5.6}$$

La variabile x rappresenta il numero di nodi percepiti dal dispositivo e $x \in \mathbb{R}, x \geqslant 0$, mentre z rappresenta il livello di batteria e $z \in \mathbb{R}, z \in [0; 100]$.

In figura 5.4 è riportato su grafico la curva del DF con l'aggiunta del Fattore di Correzione, di uno dei possibili Fattore

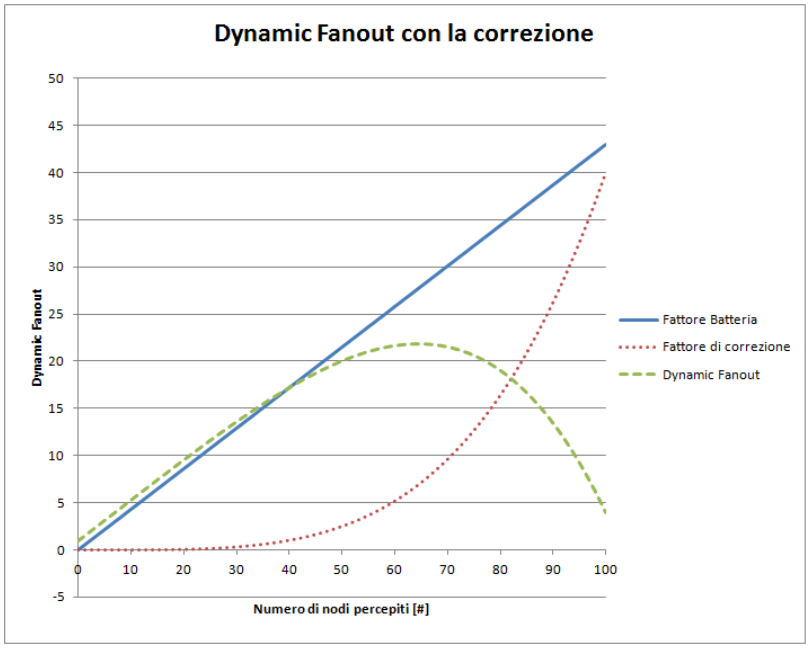


Figura 5.4: Andamento Dynamic Fanout con correzione.

Batteria. Ciò che si ottiene come DF è un andamento crescente come il FB fino all'intorno di $x = 50nodi$, poi un andamento decrescente causato dall'intervento del FC. Sottolineiamo che il valore del massimo, sia in termini di valore x e valore y, varia al variare del livello di batteria. Come è facile notare dal grafico, per più di 100 nodi o per curve di batteria inferiori, il DF risulterebbe negativo. Ovviamente questo non è accettabile. Per questo motivo abbiamo anche aggiunto una componente asintotica al calcolo del DF. L'asintoto orizzontale, è pensato per stabilizzare il DF su di un valore, quando vi sono tanti nodi. La nostra scelta è basata sull'idea che oltre un certo numero di nodi, si possa trascurare la dipendenza dalla numero di nodi percepiti in quanto ci troveremmo in uno scenario di altissima densità. Quindi abbiamo pensato che è sufficiente avere un limite asintotico dipendente dal livello della batteria in modo da garantire sempre che ogni nodo non compia sforzi eccessivi consumando troppa energia. Il valore asintotico viene preso in considerazione quando il numero di nodi supera X_max, quel valore di x corrispondete al DF_max per quella curva. Se $x > X_max$, il sistema prenderà in considerazione il valore asintotico se la curva DF con correzione è inferiore all'asintoto. Il valore dell'asintoto è stato scelto in modo da tenere la dipendenza dal livello di

batteria, così che ogni curva abbia il proprio asintoto adeguato. L'equazione dell'asintoto viene così definita:

$$Asintoto = 1 + DF_{max} \cdot 50\% \tag{5.7}$$

A questo punto abbiamo identificato due situazioni: $x < X_max$ e $x \geqslant X_max$. Unendo l'Equazione (5.6) con l'Equazione (5.7) otteniamo l'Equazione (5.8) finale del Dynamic Fanout.

$$DF = \begin{cases} 1 + \left(\frac{\sqrt{0,2 \cdot z - 1,9}}{10}\right) \cdot x - 0.0000004x^4 & se\, x < X_{max} \\ \max\left(1 + \left(\frac{\sqrt{0,2 \cdot z - 1,9}}{10}\right) \cdot x - 0.0000004x^4; 1 + \frac{1}{2}DF_{max}\right) & se\, x \geqslant X_{max} \end{cases} \tag{5.8}$$

In figura 5.5 riportiamo su grafico il risultato finale del calcolo del Dyanmic Fanout con asintoti orizzontali. Come prima, abbiamo rappresentato i dieci livelli di batteria. Dato che il DF

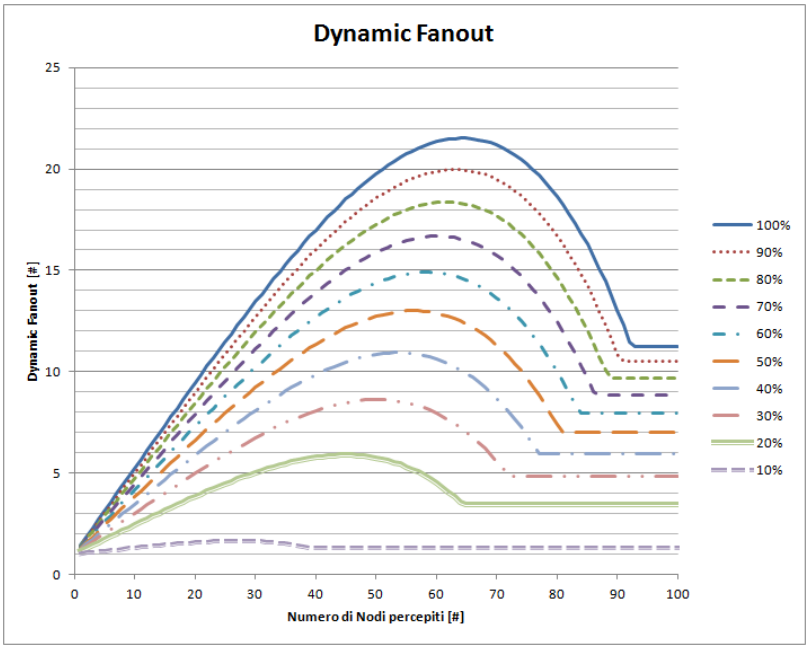


Figura 5.5: Curve finali Dynamic Fanout.

esprime il limite di trasmissioni che un nodo può fare per ogni informazione, non ha senso avere valori non interi perché il conteggio può essere fatto solo a numeri interi. Abbiamo quindi applicato un arrotondamento per eccesso alle curve mostrate in precedenza in figura 5.5. Il risultato è riportato in figura 5.6. Applicando tale arrotondamento abbiamo ottenuto un innalzamento per tutte le curve di 1; quindi avremo il valore minimo

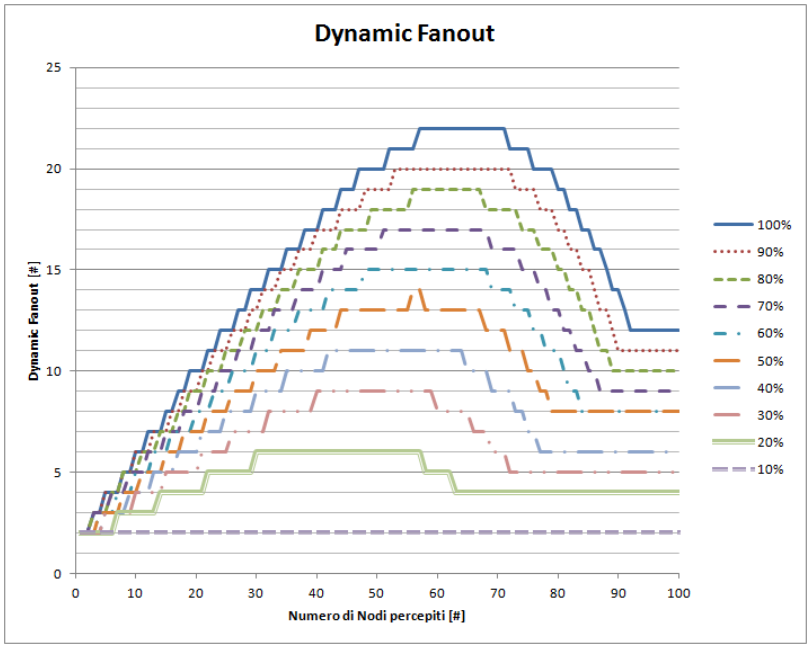


Figura 5.6: Curve finali Dynamic Fanout, arrotondato per eccesso.

pari a 2. Questo non crea alcun problema, anzi rende il sistema ancora più permissimo per reti piccole o diradate. Al crescere del numero di nodi percepiti, questo effetto resta trascurabile. Quello riportato in figura 5.6 è ciò che è stato implementato nell'algoritmo.

Abbiamo studiato anche molte altre funzioni, di vario genere e con comportamenti diversi. Alcune funzioni più conservative e altre più permissive. Di seguito riportiamo solo due casi: uno più permissivo e uno più conservativo rispetto alla soluzione scelta, con le relative formule delle funzioni utilizzate. Abbiamo riportato poi, solo i grafici finali, quelli degli andamenti totali del DF con asintoti senza arrotondamenti, per comprendere meglio le caratteristiche delle curve. Il metodo di calcolo del DF totale è lo stesso dell'Equazione (5.8), cambieranno solo le funzioni che lo compongono.

Partiamo dai Fattori Batteria. Di seguito riportiamo le due funzioni in esame, più la funzione usata in soluzione, per fare meglio un confronto. Le funzioni studiate sono state molte, ma riportiamo solo quelle generate da una manipolazione della funzione utilizzata in soluzione. In figura 5.7, sono riportate le rispettive curve per un confronto grafico.

$$y = \frac{\sqrt{0,2 \cdot x - 1,9}}{10} \tag{5.9}$$

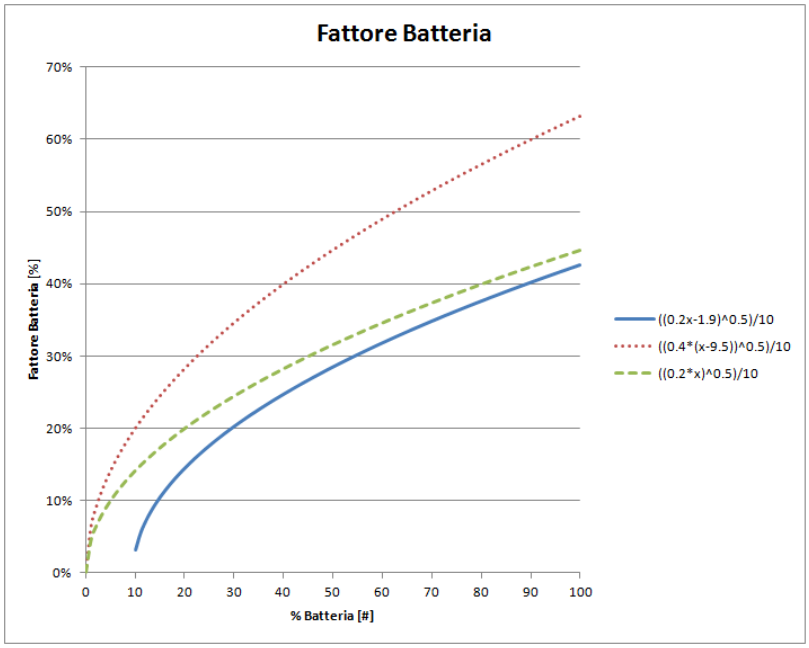


Figura 5.7: Funzioni alternative per il Fattore Batteria.

$$y = \frac{\sqrt{0,4x}}{10} \tag{5.10}$$

$$y = \frac{\sqrt{0,2x}}{10} \tag{5.11}$$

L'Equazione (5.9)è la stessa equazione della soluzione (Equazione (5.4)), mentre l'Equazione (5.10) e l'Equazione (5.11) sono rispettivamente la funzione permissiva e la funzione conservativa.

Caso Permissivo

Il caso permissivo nasce dalla rimozione del fattore di traslazione orizzontale e dall'incremento del coefficiente della variabile x, da 0.2 è stato alzato a 0.4. Questo gli permette di crescere più velocemente, come si può vedere in figura 5.7.

Il Fattore Batteria quindi è:

$$FB_{perm} = \frac{\sqrt{0,4x}}{10}$$

Aumentando il coefficiente si ottiene una crescita più ripida, quindi una reattività maggiore. Avere un fattore di partizionamento molto alto può giovare alle prestazioni, ma fino ad un

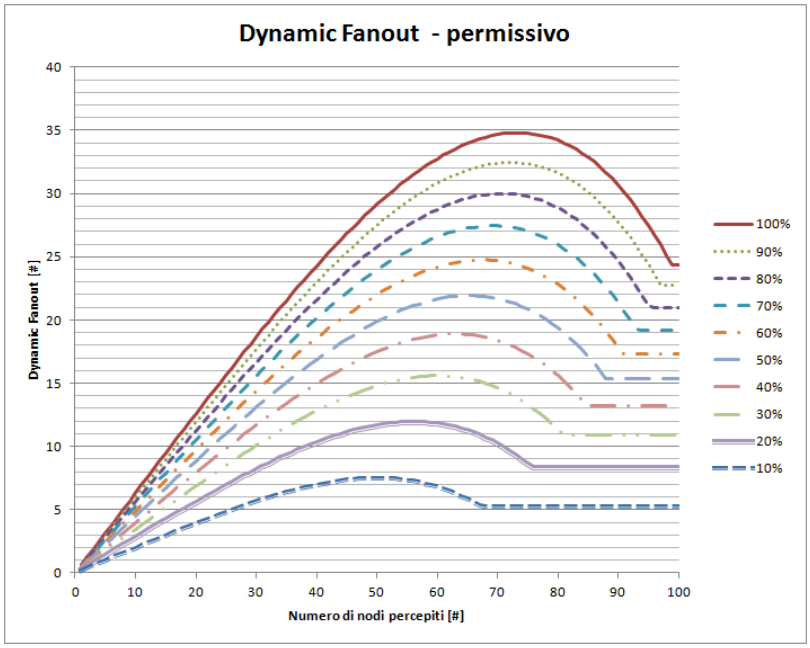


Figura 5.8: Curve DF permissivo.

certo punto perché alla fine il DF sarà così alto per ogni situazione di nodi percepiti che tale limite non sarà mai raggiunto e quindi si userà come criterio di terminazione solo l'AL, facendo diventare il DF inutile.

Il Fattore di Correzione rimane lo stesso dell'Equazione (5.5):

$$FC = 0.0000004x^4$$

L'asintoto invece è stato alzato al 70% del DF massimo:

$$Asintoto = DF_{max} \cdot 70\% \tag{5.12}$$

Il calcolo del DF rimane lo stesso della soluzione descritta in precedenza. Il risultato finale lo riportiamo nel grafico in figura 5.8. Dal grafico si può notare che abbiamo ottenuto un incremento del valore massimo di circa 60% o addirittura superiore, dipendente dalla curva.

Caso Conservativo

Per il caso conservativo, presentiamo una funzione molto simile a quella utilizzata nella soluzione, l'Equazione (5.11). E' stato rimosso il fattore di traslazione, ma il coefficiente sulle x è rimasto lo stesso. Infatti come si può vedere nel grafico in figura 5.7, la funzione conservativa e quella della soluzione hanno un andamento molto simile. La principale differenza di questa

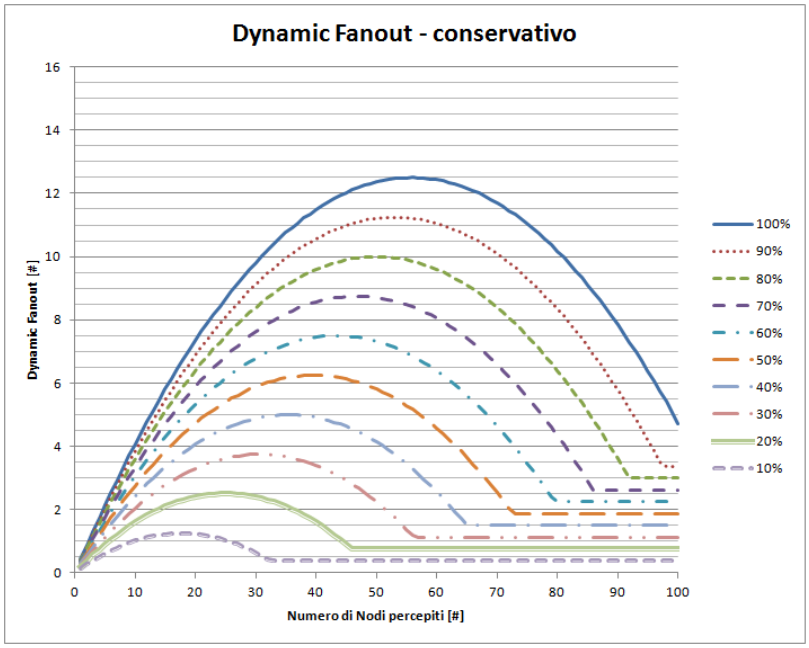


Figura 5.9: Curve DF conservativo.

versione, è di avere un Fattore di Correzione più aggressivo che agisce prima, rispetto a quello usato nelle precedenti.

$$FB_{cons} = \frac{\sqrt{0,2x}}{10}$$

Il Fattore di Correzione è cambiato, il suo coefficiente è stato aumentato e il grado della funzione diminuito, per rendere tangibile la sua correzione molto prima.

$$FC = 0.004x^2 \tag{5.13}$$

Anche l'asintoto è stato cambiato e scelto più basso, solo il 30% del DF massimo delle curve.

$$Asintoto = DF_{max} \cdot 30\% \tag{5.14}$$

Il calcolo del DF è rimasto lo stesso di quello presentato precedentemente per la soluzione. Abbiamo riportato il risultato su grafico, in figura 5.9. Come si può notare dal grafico, abbiamo ottenuto una diminuzione del valore dei massimi da un 40% a un 50% rispetto alla soluzione implementata.

5.1.4 *Advertising Limit (AL)*

L'Advertising Limit ha il compito di segnalare al dispositivo quando smettere di pubblicizzare un'informazione, perché mol-

to probabilmente è inutile e solo uno spreco di energia. Durante la fase di advertising, vengono conteggiati gli Advertising Event (AE) che vanno a vuoto, ovvero che non ricevono risposta. Se questo numero di advertising consecutivi senza successo diventa uguale o maggiore dell'AL, significa che con molta probabilità nessuno dei nodi vicini all'advertiser è interessato a quella informazione. A causa dei tempi di attesa imposti dalla tecnologia di trasmissione, non si può avere una certezza matematica, ma avendo inserito un ritardo tra un tentativo di advertising e il successivo, un nodo può essere sufficientemente sicuro. Nel caso un nodo prenda una decisione errata, significa che vi sono tanti nodi nelle vicinanze e quindi l'alta congestione ha causato una interpretazione sbagliata della situazione. Ma la stessa alta densità di nodi, fa si che la mancanza di diffusione da parte di un nodo venga coperta da altri nodi. Il valore limite è calcolato in modo dinamico, dipendente solamente dal numero di nodi che il dispositivo riesce a percepire, quindi al numero di nodi cui può potenzialmente connettersi. La dipendenza dal livello della batteria è neutra, non esegue un partizionamento come nel caso del DF. Questo perché la richiesta energetica di un singolo messaggio di advertising è molto bassa, tale da essere trascurabile rispetto al consumo energetico medio richiesto per la trasmissione di un'informazione o al consumo medio del dispositivo stesso. Continuare all'infinito a trasmettere qualcosa, anche se richiede poca energia, si traduce in un considerevole consumo. L'obiettivo di questo parametro è di far capire al dispositivo quanto i nodi intorno a lui non sono più interessati all'informazione che esso ha da trasmettere e che quindi può fermarsi e mettersi in ascolto di eventuali nuove informazioni.

Abbiamo ricercato delle funzioni con cui modellare l'Advertising Limit, che potessero dare a questo parametro il comportamento desiderato. Allo stesso modo del DF, abbiamo visto la necessità di progettare il comportamento dell'AL, tale che abbia una veloce reattività per valori piccoli di numero di nodi, mentre una crescita lenta per valori medio - grandi. Dato che il processo di advertising richiede poca energia, non abbiamo trovato la necessità di cercare forti comportamenti conservativi per valori grandi di nodi percepiti, ma lasciando la funzione col suo comportamento di crescita normale, se pur lentamente crescente. Questo perché vi sono dei timeout di attesa connessione da parte di chi sta richiedendo l'informazione. Quando un dispositivo pubblicizza un'informazione, tutti i nodi che ricevono la pubblicità e non hanno l'informazione ne faranno ri-

chiesta, ma solo il primo richiedente sarà accontentato mentre tutti gli altri staranno in attesa di connessione finché il rispettivo timeout scade. In altre parole mentre un dispositivo è in attesa di connessione, non può sapere se il mittente ha scelto lui come destinatario, l'unica cosa che può fare è aspettare che arrivi il messaggio entro il timeout di connessione; se ciò avviene, viene iniziata la trasmissione dei dati, altrimenti il nodo in attesa torna in stato di Initiating e si mette in ascolto per altri pacchetti di advertising. Durante l'attesa per il timeout, il sistema rimane "bloccato" sul possibile mittente ed ignora qualsiasi altra comunicazione come altre pubblicità della stessa informazione fatte da altri nodi. Ci siamo resi subito conto che un solo evento di advertising andato a vuoto non indica con sufficiente certezza che tutti, o quasi, i nodi attorno al dispositivo sono contagiati. Per questo motivo abbiamo progettato l'Advertising Limit senza grosse aspetti conservativi, in modo che il valore degli AE a vuoto consecutivi, sia sufficientemente alto per indicare con buona probabilità una situazione di alto contagio dei nodi vicini.

Abbiamo valutato anche per l'AL più funzioni, dai comportamenti più o meno permissivi. Le principali funzioni che abbiamo analizzato sono le seguenti:

$$y = \ln(2x) + 1 \tag{5.15}$$

$$y = \frac{\ln(2x) + \log(2x) + 2}{2} \tag{5.16}$$

$$y = \sqrt{x} + 1 \tag{5.17}$$

$$y = 2 \cdot \ln(x) + 1 \tag{5.18}$$

Abbiamo riportato su grafico le funzioni appena elencate, nella figura 5.10. Come si può notare anche dai grafici, le funzioni analizzate hanno diversi andamenti e crescono con rapidità diverse. L'Equazione (5.15) è stata la prima funzione studiata, in quanto di base ha le caratteristiche che cercavamo; un andamento reattivo per valori bassi di nodi e un comportamento crescente ma molto più limitato al crescere del numero di nodi. L'Equazione (5.16) è la media tra due funzioni logaritmiche e si è rivelata essere abbastanza conservativa. Le Equazioni (5.17) e (5.18) invece sono due funzioni molto simili come

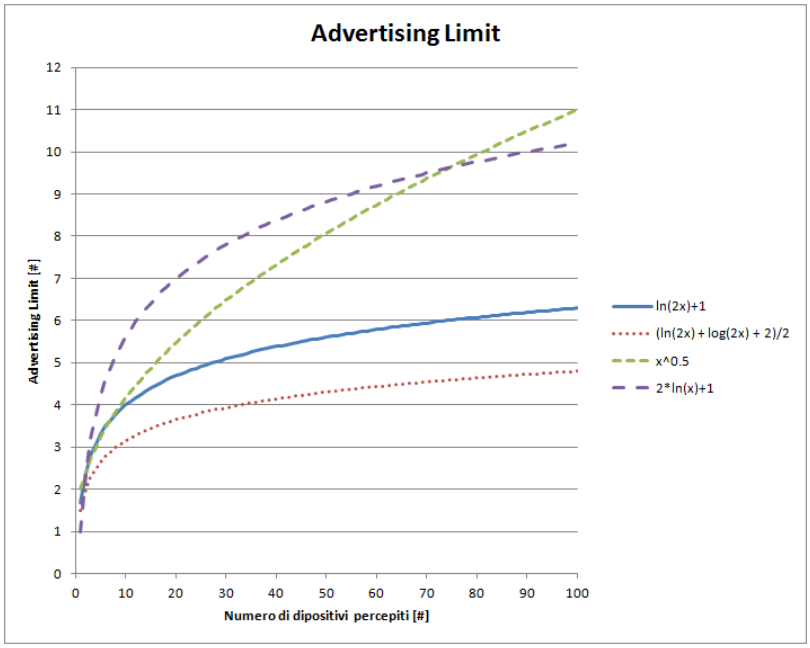


Figura 5.10: Curve di Advertising Limit.

andamento ed entrambe molto permissive, con una crescita abbastanza rapida e costante. La funzione che abbiamo scelto di implementare nell'algoritmo è la funzione di Equazione (5.15):

$$AL = \ln(2x) + 1$$

L'idea generale nel modellare questo parametro è stata quella di cercare di dare reattività nell'adattamento al crescere del numero di nodi percepiti, quando i nodi percepiti sono pochi. La rapidità di crescita per valori piccoli di numero di nodi, è per poter dare ai dispositivi un limite sufficientemente ridondante che possa coprire gli eventuali nodi che con poca batteria parteciperanno poco, o nulla, alla diffusione dell'informazione. Più il numero di nodi cresce più la necessità di questa copertura diminuisce, rendendo sufficiente una crescita dell'AL più lenta; questo perché vi sono molti più nodi che possono coprire le eventuali mancanze e queste coperture si distribuiscono tra tutti i nodi. In figura 5.11 abbiamo riportato su grafico l'arrotondamento per eccesso delle funzioni analizzate riportate nel grafico in figura 5.10, per lo stesso motivo esposto per il DF. La versione con arrotondamento implementato nell'algoritmo.

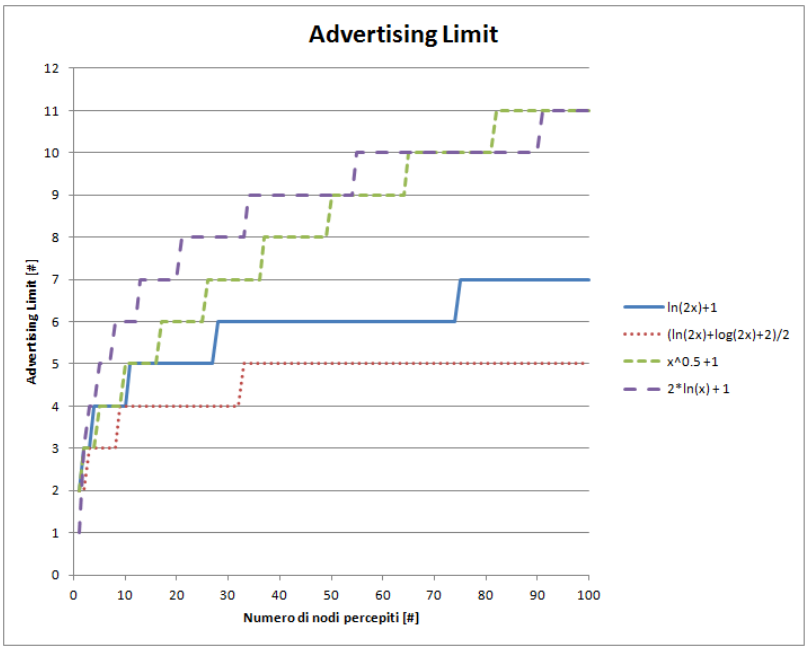


Figura 5.11: Curve di Advertising Limit, arrotondate per eccesso.

6

SIMULAZIONI E VALUTAZIONE RISULTATI

6.1 ORGANIZZAZIONE SIMULAZIONI

La parte di realizzazione sperimentale del nostro progetto, è stata fatta tramite simulazioni con lo scopo di verificare il comportamento del nostro algoritmo e raccogliere dati per fare un'analisi più oggettiva delle sue prestazioni al variare delle condizioni. Per avere una sufficiente base statistica, abbiamo impostato nel file d'inizializzazione il numero di simulazioni a 20. Significa che per ogni configurazione il simulatore eseguirà venti simulazioni consecutive senza reimpostare il generatore di numeri casuali, in questo modo abbiamo ottenuto venti configurazioni di rete differenti per ogni configurazione. Per impostare il numero di ripetizioni da eseguire, abbiamo inserito nella sezione General di ogni file d'inizializzazione utilizzato, l'istruzione "repeat = 20".

Le simulazioni sono state organizzate rispetto a due parametri principalmente: la densità di nodi e il raggio di trasmissione del BT. Come presentato nella Sezione 5.1, le densità studiate sono:

- $d = 0.02 \frac{nodi}{m^2}$.
- $d = 0.01 \frac{nodi}{m^2}$.
- $d = 0.008 \frac{nodi}{m^2}$.
- $d = 0.001 \frac{nodi}{m^2}$.
- $d = 0.0005 \frac{nodi}{m^2}$.
- $d = 0.0001 \frac{nodi}{m^2}$.

E i valori del raggio ρ sono:

- ρ = 15 m.
- ρ = 50 m.

Le densità da $d = 0.02\,nodi/m^2$ a $d = 0.001\,nodi/m^2$ sono state pensate per simulare ambienti urbani che possono essere dal mediamente popolati al densamente popolati. Situazioni simili sono il caso di medie-grandi città o per le densità più elevate, situazioni di forte concentrazione di persone in un'area ristretta. Quest'ultimo caso può essere generato di una locale alta concentrazione abitativa come una serie di palazzi o condomini vicini tra loro, ma anche da motivazioni esterne come eventi di qualunque genere. Le densità più piccole, $d = 0.0005\,nodi/m^2$ e $d = 0.0001\,nodi/m^2$ sono state pensate per studiare la scalabilità del sistema a fronte della dispersione dei nodi su una vasta area, ma anche per simulare situazioni urbanistiche tipiche di piccoli comuni, con un relativo basso numero di abitanti come ad esempio i piccoli paesi di campagna. Abbiamo quindi organizzato il processo di realizzazione sperimentale in modo da eseguire, per ogni densità, simulazioni per entrambi i raggi ρ e collezionare dati per un'analisi a posteriori delle performance.

Il principale risultato che abbiamo voluto monitorare è stato la percentuale di diffusione dell'informazione al diminuire della densità, per studiare quali fossero i limiti di applicabilità in termini di efficienza del sistema e per avere un insieme di scenari in cui l'algoritmo da noi proposto presentasse buoni livelli di prestazioni. Parallelamente abbiamo anche raccolto dati sul tempo totale di trasmissione, inteso come l'istante di tempo in cui l'ultima trasmissione si è conclusa. Abbiamo scelto di raccogliere anche quest'ultimo dato perché abbiamo cercato di capire con che tempistica il messaggio è diffuso. Il motivo è stato di voler dare un corrispettivo valore temporale alla percentuale di nodi che l'algoritmo raggiunge. Da solo un valore di tempo massimo di trasmissione non è valutabile, poiché è molto dipendente dalla dispersione spaziale della rete; per come è costruita la rete non vi è garanzia che tutti i nodi siano connessi ad uno stesso grafo. Per questo motivo, il tempo totale di trasmissione deve essere analizzato in coppia con il valore percentuale di rete coperta durante quella specifica simulazione. In questo modo possiamo avere un'idea di come il tempo di propagazione evolva al variare della copertura e della densità della rete.

6.1.1 *Algoritmo Dynamic Fanout: funzionamento*

In questa sezione diamo una breve spiegazione sul funzionamento dell'algoritmo all'atto pratico dell'esecuzione. Presenteremo il suo comportamento descrivendo le tre principali configurazioni di lavoro. Faremo riferimento ai diagrammi di flusso in Appendice A durante la descrizione. Ricordiamo che i diagrammi rappresentano l'elenco di istruzioni della relativa situazione di lavoro. I tre principali momenti di lavoro sono i seguenti: standby, invio di un nuovo messaggio e Ricezione di un messaggio. Vi è anche un'insieme di istruzioni parallelo alle tre configurazioni, che viene eseguito sempre, in maniera periodica.

Standby

Partiamo con la situazione di standby e si faccia riferimento al diagramma in Appendice A.1.1. Quando il sistema si trova in una situazione di Standby, vuol dire che in quel momento non è impegnato in nessuna trasmissione nè ricezione. In questa situazione di lavoro il sistema, inizia nello stato di Standby del BLE. All'occorrenza dell'esecuzione delle azioni periodiche, vengono eseguiti i rispettivi calcoli periodici e poi viene fatto un controllo se la batteria è sopra la soglia limite del 10% e se il sistema non sia occupato. Ovviamente essendo in stato di Standby il sistema non è impegnato in nessuna attività e se vi è sufficiente batteria, il sistema passa in stato di Initiating, rimanendo in ascolto e aspettando eventuali comunicazioni sulla rete. Parallelamente, con cadenza periodica verranno eseguire le azioni periodiche e ogni volta, dopo, verrà fatto il controllo sul livello di batteria. Qualora esso scenda sotto la soglia limite, il sistema verrà portato in stato di Standby e da li non si muoverà finché la batteria non verrà ricaricata. Le azioni periodiche vengono eseguite sempre anche quando il sistema in Standby per monitorare lo stato della rete e della batteria del dispositivo.

Indipendentemente dallo stato in cui il sistema si trova, Standby o Initiating, se il sistema riceve dall'utente il comando di inviare un nuovo messaggio, il dispositivo passa a eseguire le istruzioni della configurazione di invio messaggio. Viene lasciata la possibilità di inviare un messaggio dallo stato di Standby, anche se la batteria è inferiore al 10%, perché negli scenari di lavoro ipotizzati, potrebbe rivelarsi comunque utile inviare un nuovo messaggio, anche se si potrebbe inficiare sull'autonomia del dispositivo. In tale situazione il sistema invierà solo una volta il messaggio e poi si riporterà da solo in stato di Standby.

Nel caso invece il sistema si trovi in stato di Initiating e percepisca un nuovo messaggio sulla rete, si porta in configurazione di ricezione.

Azioni periodiche

Sempre in riferimento a Appendice A.1.1, descriviamo cosa trattano le azioni periodiche. Il sistema schedula periodicamente un insieme di istruzioni da eseguire, chiamate azioni periodiche. Queste azioni sono eseguite sempre, indipendentemente dalla situazione e dalla configurazione in cui il dispositivo sta operando. La periodicità è stata scelta di 30 secondi. Le azioni periodiche servono per aggiornare i parametri utilizzati dall'algoritmo di diffusione (DF e AL), in modo che dinamicamente si adattino all'ambiente esterno e allo stato attuale del dispositivo stesso. Dopo le operazioni di aggiornamento, viene anche fatto un controllo sul livello della batteria, come descritto per la configurazione di standby. Il controllo riguarda il livello della batteria e se il sistema rileva che la batteria è sopra la soglia limite (10% come soglia limite), allora non interviene, ma nel caso sia sotto la soglia limite, viene segnalato e se il dispositivo non è impegnato il sistema viene portato in stato di Standby in cui rimarrà finché la batteria non tornerà sopra il 10%. Questi controlli avvengono anche durante trasmissioni di scambio dati, ma in quel caso il sistema è occupato e si lascia terminare l'operazione, prima di portare il sistema in Standby. Non si vuole che una trasmissione già iniziata venga troncata a causa del controllo, poiché non vogliamo che vada a intralciare il processo di diffusione dell'informazione, ma può comunque agire tra una trasmissione e l'altra. Questo non grava troppo sull'autonomia in quanto il singolo scambio dati, per quanto grossa possa essere l'informazione, non rappresenta una grossa richiesta energetica.

Anche quando la batteria è sotto la soglia limite e il sistema è in stato di Standby, i controlli, grazie alla loro esigua richiesta energetica, sono sempre effettuati per riabilitare il sistema al lavoro attivo qualora la batteria venga ricaricata e torni sopra soglia.

Una parte dei controlli periodici è già stata inserita e rappresentati nel diagramma di flusso della configurazione di standby, per dare una semplice idea di quale sia il compito di questi controlli, ma non sono vincolati a nessuna configurazione. Esse restano organizzate ed eseguite in maniera indipendente.

Per quanto riguarda la sola parte di simulazione, abbiamo inserito nelle azioni periodiche anche il decremento della bat-

teria dei dispositivi, per simulare l'uso del dispositivo da parte dell'utente. Il decremento è piccolo per indicare il consumo medio in idle. Per le trasmissioni completate invece, vi è un decremento più grande dovuto alla grandezza dell'informazione.

Invio di un messaggio

Quando è necessario inviare un nuovo messaggio, che esso sia stato ricevuto dalla rete oppure venga inviato dall'utente, il sistema entra in configurazione di invio. Si faccia riferimento al diagramma all'Appendice A.1.2. Appena entrato in questa modalità, il sistema controlla di eseguire una transizione di stato corretta, quindi se non era già in stato di Standby, si porta in stato di Standby. Poi vengono inizializzati i contatori di trasmissioni e di AE. A questo punto può entrare in stato di Advertising e iniziare le operazioni per l'invio. Viene incrementato di uno il contatore degli AE e poi il sistema esegue l'AE vero e proprio. Durante l'AE il sistema trasmette pacchetti di tipo "advertising indiretto" (ADV_IND [8]). Dopo di che si mette in ascolto, aspettando eventuali risposte e viene fatto partire il timeout. Se il timeout scade e nessuna richiesta di connessione è arrivata al dispositivo, esso controlla se il *AE counter* è minore dell'AL. In caso positivo, vuol dire che nessuno è più interessato e quindi viene riportato il sistema in configurazione di standby. In caso negativo invece, il sistema ricomincia tutto il processo di advertising.

Nel caso invece che, prima dello scadere del timeout, arrivi una richiesta di connessione (CONN_REQ [8]), il sistema procede a segnalare che è occupato in una trasmissione, così da evitare interruzioni dai controlli di sistema, e poi entra in stato di Connection col ruolo di Slave. Quando il nodo col ruolo di Master invia la richiesta id *pool*, inizia il vero e proprio Connection Event (CE), in cui il sistema trasmette al richiedente tutta l'informazione. Al termine del CE, viene incrementato il contatore di trasmissioni effettuate, viene decrementato il livello della batteria a causa della trasmissione appena effettuata, il sistema segnala di non essere più occupato, passa in stato di Standby e controlla che se ha raggiunto il limite di trasmissioni imposto dal DF. In caso affermativo, il sistema ha raggiunto la sua quota di lavoro e può tornare in configurazione di standby, mentre in caso contrario, il sistema si riporta in stato di Advertising pronto a ricominciare.

Ricezione di un messaggio

Quando il dispositivo rileva che uno o più nodi stanno diffondendo informazioni che il sistema non ha ancora ricevuto, esso entra in configurazione di ricezione di un messaggio, descritto dal diagramma in Appendice A.1.3. Questa configurazione è accessibile solo da uno stato di ascolto, ma comunque viene controllato come prima cosa che il sistema sia nel corretto stato. Se non si trova in stato di Initiating, si porta prima in stato di Standby e poi in stato di Initiating. A questo punto il sistema è pronto alla connettersi, quindi risponderà al primo pacchetto di advertising (ADV_IND [8]) che riceve, inviando a sua volta una richiesta di connessione (CONN_REQ [8]). Dopo di che, il sistema si segna occupato e entra in stato di Connection col ruolo di Master. A questo punto invia la sua richiesta di *pool* e lancia il timeout. Il sistema non sa ancora se il dispositivo mittente ha accettato la sua connessione. L'unica cosa che può fare dopo aver inviato la richiesta di pool, è aspettare. Se il timeout scade e non è stata ricevuta nessuna informazione, vuol dire che il mittente non ha accettato la richiesta di connessione oppure era già impegnato in un altra trasmissione. Capendo di non essere stato scelto per la trasmissione, il sistema si dichiara non occupato e si riporta in stato di Initiating in ascolto, pronto a ricominciare.

Nel caso invece la richiesta di connessione venga accetta, in risposta alla richiesta di *pool*, inizierà il CE e il dispositivo advertiser invierà l'informazione. Una volta terminato il CE, viene decrementato il livello della batteria a causa della trasmissione appena effettuata, il sistema si dice non più occupato e si prepara ad entrare in configurazione di invio per diffondere a sua volta l'informazione.

6.2 VALUTAZIONE RISULTATI

Dopo aver terminato le simulazioni, abbiamo raccolto i dati ed eseguito analisi su di essi. I parametri che abbiamo monitorato come risultati, a fronte dell'invio di un singolo messaggio sono stati:

- Copertura: percentuale di contagio che il messaggio ha raggiunto.
- Tempo totale di trasmissione: tempo totale impiegato per raggiungere tale copertura di rete.

Grazie alle funzionalità offerte dal framework di simulazione OMNeT++, è stato possibile raccogliere i dati in maniera efficiente. Con una manipolazione dei dati appena raccolti, il framework è stato in grado di fornirci il valor medio, la varianza, la deviazione standard e anche l'intervallo di confidenza al 95%. Abbiamo anche estratto dai dati raccolti, gli stessi indici tramite un foglio di calcolo Excel, in modo da poter validare ed eventualmente correggere eventuali errori o approssimazioni del software di simulazione.

Prima di parlare dei singoli risultati, ripetiamo alcuni aspetti che hanno caratterizzato l'algoritmo, i modelli e la simulazione stessa. Per avere una maggior variabilità nella simulazione, abbiamo assegnato molti valori di parametri tramite l'uso di generatori casuali, come ad esempio la disposizione dei nodi nella rete, il livello di batteria iniziale di ogni dispositivo e, per ogni ripetizione, la scelta del nodo che possiede l'informazione da divulgare e che inizierà la diffusione. Per ogni densità, sono stati simulati i due range ρ scelti, $\rho = 15m$ e $\rho = 50m$, e poi è stata fatta la media tra i due. Inoltre la diversità di prestazioni dovuta alla differenza di raggio ρ utilizzato è elevata. Questi aspetti saranno ripresi quando illustreremo i risultati ottenuti.

Il primo parametro che presentiamo è quello della percentuale di rete contagiata dal messaggio, la ***copertura***. In figura 6.1 sono riportati su grafico i valori medi dei dati raccolti relativi alla copertura raggiunta dal messaggio, per ogni densità simulata. Per ogni curva abbiamo anche riportato sul grafico, il suo intervallo di confidenza al 95%, per comprendere la distribuzione attorno ai valori medi. La prima cosa che si nota è che per le densità $d = 0.02\,nodi/m^2$, $d = 0.01\,nodi/m^2$ e $d = 0.008\,nodi/m^2$ la percentuale di contagio è intorno al 100% ed è indicativo di un'ottima efficienza prestazionale della soluzione da noi proposta in ambienti densamente popolati o in momenti di alta concentrazione urbana per motivi particolari. Per queste tre densità molto alte entrambi i raggi ρ hanno dato ottimi valori. Le tre densità più piccole invece hanno evidenziato i limiti di operabilità del sistema. Per la densità $d = 0.001\,nodi/m^2$, abbiamo ottenuto ancora degli ottimi valori in termini di copertura, ma in questo caso, il raggio $\rho = 15m$ ha cominciato a evidenziare problemi nello stabilire collegamenti tra i nodi e quindi raggiungere un'ottima copertura. Per questo motivo abbiamo ottenuto un intervallo di confidenza già ben più ampio rispetto alle tre casistiche precedenti nelle quali era pressoché nullo o molto piccolo. Nel complesso rimane comun-

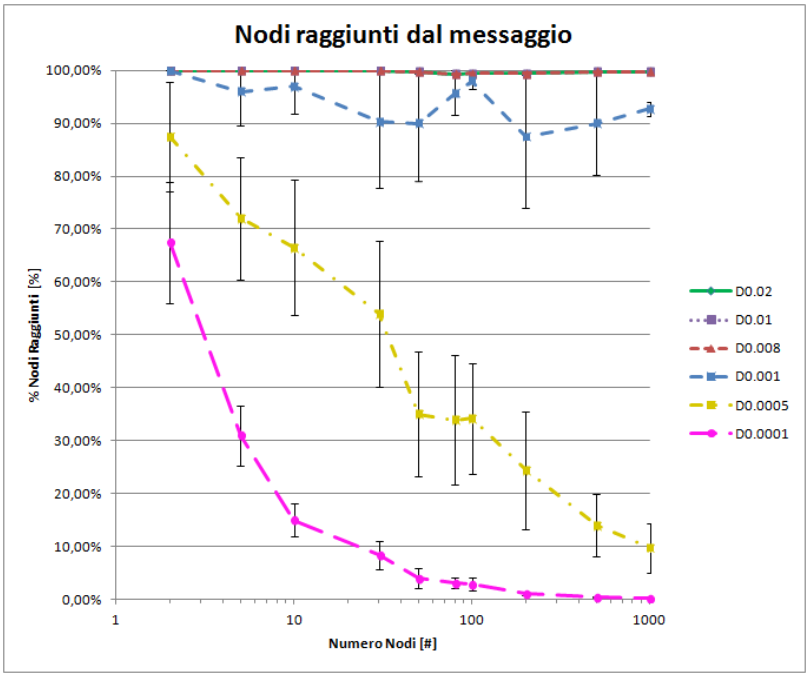


Figura 6.1: Grafico della copertura della rete.

que stabile in un intorno del 90% al crescere del numero dei nodi nella rete.

Poi abbiamo la curva della densità $d = 0.0005\,nodi/m^2$. Questo valore di densità, rispetto al precedente, presenta un andamento decrescente, al crescere del numero di nodi. Già a questo livello di densità, abbiamo riscontrato che al crescere della rete, aumenta la probabilità che si formino sottoreti isolate che causano solo una diffusione parziale del messaggio. Rispetto alla densità precedente che era il doppio, abbiamo avuto un forte deterioramento nella copertura della rete e un più ampio intervallo di confidenza. Con questo valore di densità, si ha che al crescere del numero di nodi, essi comincino ad essere troppo diradati nella rete e nel complesso si vengano a creare sottoreti sempre più piccole col crescere del numero di nodi. Con le ipotesi fatte, non vi è modo per l'algoritmo di eludere o superare quest'ostacolo. Nonostante quest'andamento decrescente, possiamo vedere che con mille nodi nella rete, il sistema sarebbe in grado di coprire una porzione di rete intorno al 10%; per questo motivo la densità $d = 0.0005\,nodi/m^2$ può essere considerata un limite inferiore di operatività. Infine l'ultima densità $d = 0.0001\,nodi/m^2$ si dimostra essere oltre le possibilità operative del sistema, infatti, mostra un degrado prestazionale forte già con reti piccole composte di soli dieci nodi. Dalle nostre si-

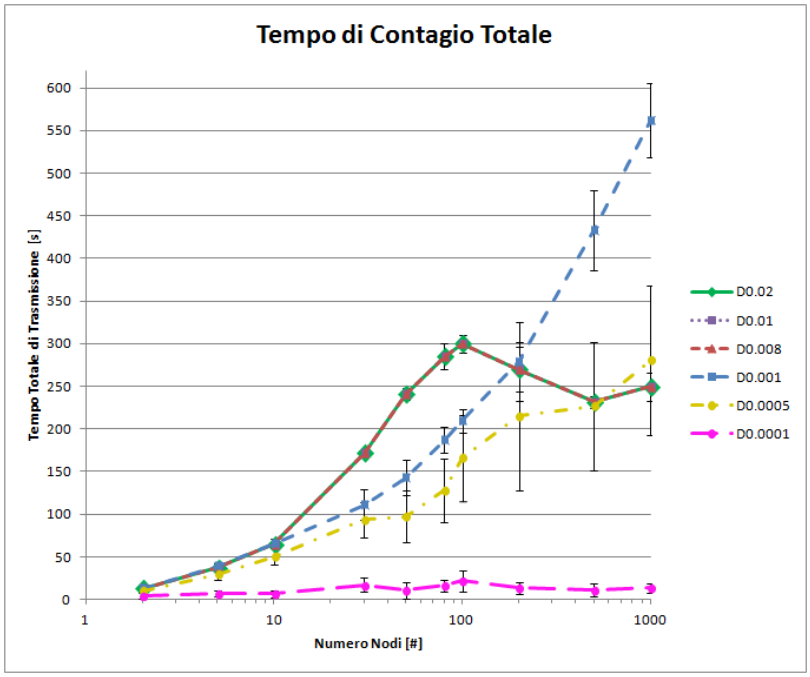


Figura 6.2: Grafico del tempo totale di trasmissione.

mulazioni è emerso che il sistema è in grado di gestire e operare a livelli ottimi, copertura intorno al 90% circa, fino a densità $d = 0.001\,nodi/m^2$ ma può operare, se pur con un degrado prestazionale linearmente dipendente dalla grandezza della rete, anche fino a una densità minima di $d = 0.0005\,nodi/m^2$.

In figura 6.2 riportiamo i risultati riguardanti il ***tempo totale di trasmissione***, tempo che l'algoritmo ha impiegato per completare l'ultima trasmissione di contagio a lui possibile. Dalle nostre analisi è risultato che per densità basse come $d = 0.02\,nodi/m^2$, $d = 0.01\,nodi/m^2$ e $d = 0.008\,nodi/m^2$, l'alta vicinanza a permesso di contenere il tempo necessario e dopo un picco in corrispondenza di cento nodi, esso diminuisce pur garantendo una copertura totale o quasi. Questo grazie all'alta ridondanza di collegamenti creatasi dalla vicinanza dei nodi; in questo modo la rete stessa riesce a evitare la formazione di colli di bottiglia tra gruppi di nodi. Per $d = 0.001\,nodi/m^2$ invece, abbiamo un andamento crescente col numero di nodi, anche se in figura 6.1 si vede come resti sempre vicino alla copertura totale. In questo caso abbiamo un andamento di tipo quadratico, al crescere del numero di nodi nella rete. Già passando da $d = 0.008\,nodi/m^2$ a $d = 0.001\,nodi/m^2$ si può notare come cambi il tempo necessario all'algoritmo per raggiungere la sua massima copertura. Infine, le densità d=0.0005 e d=0.0001 presentano curve inferiori alle precedenti ma vanno

lette in combinazione con i risultati mostrati in figura 6.1. Notiamo che l'andamento, almeno per la curva $d = 0.0005\,nodi/m^2$, riamane di tipo quadratico, indice di una rete sempre più diradata nella quale si perdono sempre più i collegamenti ridondanti tra i nodi e compaiono sempre più sottoreti e singoli collegamenti tra essi, che sono il punto debole dell'algoritmo. Se al crescere del numero di nodi, la percentuale di rete coperta diminuisce, significa che il sistema esegue sempre meno trasmissioni, per questa ragione le curve per $d = 0.0005\,nodi/m^2$ e $d = 0.0001\,nodi/m^2$ sono inferiori alle altre. Per la curva di densità più bassa $d = 0.0001\,nodi/m^2$, possiamo solo da dire che conferma i dati rilevati della copertura. L'impossibilità eseguire trasmissioni a causa di una rete troppo sparsa, fa sì che l'algoritmo perda efficacia e si fermi dopo pochi secondi di operatività.

Per valutare meglio l'efficacia della propagazione, abbiamo anche analizzato un ***fattore di efficienza***: il rapporto tra la copertura raggiunta, in termini di numero di nodi, e il tempo impiegato per raggiungerla. In figura 6.3 riportiamo i risultati su grafico. Dal grafico si nota come tutte le densità tra $d = 0.02\,nodi/m^2$ a $d = 0.001\,nodi/m^2$ siano molto simili in termini di efficienza. L'estrema vicinanza tra i nodi per $d = 0.02\,nodi/m^2$, rende l'algoritmo molto efficace, ma anche scendendo fino a $d = 0.001\,nodi/m^2$ l'efficienza dell'algoritmo rimane buona e l'andamento rimane lo stesso delle densità più alte. Per le due densità limite invece si nota come l'efficienza per la $d = 0.0005\,nodi/m^2$ fatichi ad aumentare all'aumentare del numero di nodi, mentre per la $d = 0.0001\,nodi/m^2$ si noti come continui a decrescere, fatta eccezione per qualche piccolo picco e per il valore superiore per $n = 2$. Quest'ultimo fatto è dovuto a una scelta di progettazione, per agevolare l'analisi dei dati. Quando il primo nodo è inizializzato col messaggio da inviare, il sistema registra il tempo di consegna come l'istante del primo contagio che è sicuramente inferiore al tempo necessario per contagiare l'altro nodo, in una rete formata da due soli nodi. Per questo motivo, per reti piccole risulta avere un'efficienza più alta. All'aumentare della grandezza della rete la curva per $d = 0.0001\,nodi/m^2$ si stabilizza e non presenta più il problema, rappresentando il corretto andamento.

Dopo l'analisi dei dati raccolti, è difficile poter dire con che legge si comporta il nostro algoritmo, per vari motivi. Il primo tra tutti è che non abbiamo sempre una situazione in cui ogni nodo sia connesso in un unico grafo. Le diverse densità pos-

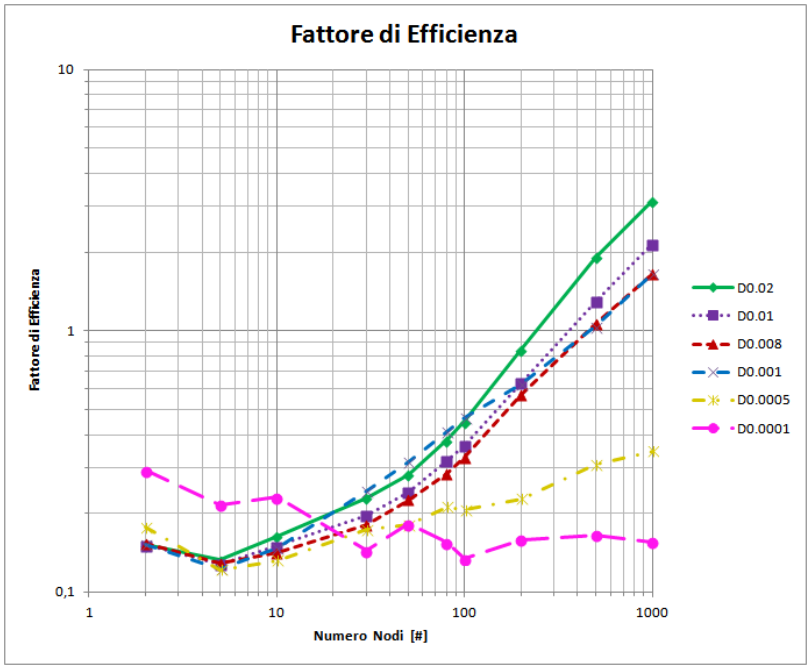


Figura 6.3: Grafico del fattore di efficienza

sono creare più sottografi isolati senza la possibilità che questi ultimi possano comunicare. Un altro fattore importante è che i nodi del grafo sono dispositivi con una batteria e l'energia a disposizione si consuma nel tempo. Nodi in posizioni critiche della rete possono esaurire la batteria e così escludere parte della rete. Sul fattore batteria sono calibrati i parametri del sistema, e i limiti di trasmissione. Un nodo con limiti di trasmissione bassi, potrebbe completare il suo lavoro di trasmissione, ma verso una zona di rete già abbastanza informata e non in una zona di rete in cui l'informazione non è ancora arrivata. Un altro fattore molto importante e influente, è il metodo (o i metodi) di terminazione che si è scelto di utilizzare. Differenti metodi con differenti criteri possono portare ad avere prestazioni molto differenti.

In condizioni ideali, tutti i nodi connessi in un unico grafo e batteria infinita per tutti i dispositivi, come presentato in [19] e in [27], gli algoritmi di rumor mongering con strategia di diffusione push&pull hanno queste caratteristiche:

- L'informazione viene diffusa a tutti dopo $O(\ln n)$ cicli.
- La copertura completa richiede almeno $O(n \cdot \log \log n)$ trasmissioni/messaggi.

Con n il numero di nodi della rete. Per gli algoritmi di rumor mongering di tipo "*address-indipendent*", le complessità ap-

pena elencate sono completamente indipendenti dal modo in cui vengono scegli i nodi con cui comunicare e le distribuzioni di probabilità con cui i nodi vengono scelti. Gli algoritmi di tipo "*address-indipendent*" sono algoritmi che, ad ogni iterazione, non dipendono dalla posizione e/o dall'indirizzo dei nodi vicini, ma solo dal numero di nodi vicini. Per questo motivo, questi algoritmi lavorano allo stesso modo sia se hanno una visione completa della rete sia se hanno una visione parziale della rete, l'importante è che siano "*address-indipendent*".

Aspetti che non sono stati considerati in questo studio e che possono avere un impatto sulle prestazioni sono: la mobilità dei nodi e una distribuzione dei nodi nell'area non più totalmente uniforme, ma con una logica più vicina al centro abitativo. Le zone abitative sono costruite vicino ad altre già presenti e non in maniera uniforme su un determinato territorio. La mobilità può creare situazioni riconducibili ai veri contagi epidemici. Un nodo a conoscenza di un'informazione può spostarsi in una zona, anche lontana dal punto di partenza, dove l'informazione sarebbe impossibilitata ad arrivare e iniziarne la diffusione. Questo porterebbe a ridurre i problemi generati dalla bassa densità e dalla conseguente formazione di sottoreti isolate.

7

DIREZIONI FUTURE

Direzioni e studi futuri si possono concentrare sul considerare la mobilità dei nodi, in quando persone; dispositivi contagiati che si spostano possono incrementare le prestazioni in termini di copertura della rete, proprio come nei casi delle epidemie. Studiare se vi sono possibilità di sfruttare maggiormente tutta la piconet e quindi poter associare più dispositivi slave per ogni master. Studiare migliori modelli di reti e migliori metodi per simulare la distribuzione abitativa dei paesi; in ogni città, gli abitanti non sono mai distribuiti uniformemente su tutta l'area sotto la giurisdizione comunale. Lo studio di eventuali pattern o di modelli più accurati, può incrementare l'efficienza anche a densità basse. Lo studio di una maggiore gestione dei messaggi, magari con l'inserimento di un TTL nei messaggi e politiche di rinvio di messaggi che erano già stati accantonati, al rilevamento di nuovi dispositivi nell'area circostante.

8

CONCLUSIONI

La nostra ricerca si pone nell'ambito dello studio per la riduzione e ottimizzazione del consumo energetico e nell'ambito dei dispositivi mobile. Il nostro lavoro si è concentrato per lo studio e la ricerca di una possibile soluzione alla mancanza delle comuni reti di comunicazioni, quali reti telefoniche e internet. La nostra ricerca ha prodotto un algoritmo dinamico, progettato come estensione di un algoritmo di gossip, che sfrutta come canale di trasmissione la tecnologia Bluetooth Low Energy, equipaggiata su tutti i più comuni dispositivi mobili in commercio. Il nostro algoritmo sfrutta le caratteristiche del gossip per diffondere informazioni ma grazie al suo dinamismo, cerca sempre di trovare un compromesso nello scegliere il carico di lavoro del dispositivo, per garantire un buon consumo energetico per non degradare troppo l'autonomia del dispositivo ma allo stesso tempo un'efficiente azione di gossip. Abbiamo analizzato il sistema per varie densità e per due diversi potenziali raggi d'azione. I risultati hanno mostrato un'ottima applicabilità fino a densità $d = 0.001\,nodi/m^2$, con una copertura nell'intorno del 90%, mentre per densità più piccole un lineare degrado delle prestazioni all'aumentare del numero di nodi nella rete, a causa della formazione di sottoreti isolate. In prima analisi, i risultati sono promettenti sia nell"ottica di futuri studi sia nell'ottica di un miglioramento delle prestazioni della tecnologia Bluetooth.

Parte I

BIBLIOGRAFIA E APPENDICE

BIBLIOGRAFIA

[1] Montresor Alberto e Jelasity Márk. «Peersim: A scalable p2p simulator». In: *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference* (2009), pp. 99–100 (cit. a p. 23).

[2] Tiberto Alessia. «Confronto tra Bluetooth Basic Rate e Bluetooth Low Energy». Bachelor Degree. institution. URL: http://tesi.cab.unipd.it/44150/1/tesi_tibertoa.pdf (cit. alle pp. 8, 49, 50).

[3] Varga Andras. «OMNeT++». In: *Modeling and Tools for Network Simulation*. Springer Berlin Heidelberg, 2010, pp. 35–59 (cit. a p. 25).

[4] Varga András. «Using the OMNeT++ discrete event simulation system in education». In: *Education, IEEE Transactions* 42 (4 ago. 2002), pp. 331–340 (cit. a p. 25).

[5] Varga András e Hornig Rudolf. «An overview of the OMNeT++ simulation environment». In: *Simutools '08* (60 2008). URL: http://dl.acm.org/citation.cfm?id=1416290 (cit. alle pp. 25, 26).

[6] *Bluetooth Basics*. Bluetooth SIG. URL: http://www.bluetooth.com/Pages/Basics.aspx (cit. a p. 8).

[7] *Bluetooth Brand*. Bluetooth SIG. URL: http://www.bluetooth.com/Pages/Bluetooth-Brand.aspx (cit. a p. 6).

[8] *Bluetooth Core Specification 4.0*. 10 Giu. 2010. URL: https://www.bluetooth.org/en-us/specification/adopted-specifications (cit. alle pp. 5, 8, 9, 34, 73, 74).

[9] *Bluetooth smart marks*. Bluetooth SIG. URL: https://www.bluetooth.org/en-us/bluetooth-brand/smart-marks (cit. a p. 5).

[10] R. Bringhurst. *The Elements of Typographic Style*. Version 3.2. Point Roberts, WA, USA: Hartley & Marks Publishers, 2008.

[11] Gomez Carles, Oller Joaquim e Paradells Josep. «Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology». In: *Sensors 2012* 12 (9 ago. 2012), pp. 1587–1611 (cit. alle pp. 8, 34).

[12] Baschieri Daniele, Corazza Michele e Gombi Stefano. *Una comparazione tra algoritmi di GOSSIP su differenti simulatori*. Report. Set. 2014. URL: http://www.doc4net.it/doc/2720361403100 (cit. a p. 20).

[13] Perez-Palacin Diego, Mirandola Raffaela e Merseguer José. «Accurate Modeling and Efficient QoS Analysis of Adaptive Systems under Bursty Workload». In: *JOURNAL OF LATEX CLASS FILES* 6.1 (gen. 2007) (cit. a p. 4).

[14] Perez-Palacin Diego, Mirandola Raffaela e Merseguer José. «On the relationships between QoS and software adaptability at thearchitectural level». In: *Journal of Systems and Software* 87 (gen. 2013), pp. 1–17 (cit. a p. 4).

[15] Perez-Palacin Diego, Mirandola Raffaela e Merseguer José. «QoS and energy management with Petri nets: A self-adaptive framework». In: *Journal of Systems and Software* 85 (12 dic. 2012), pp. 2796–2811 (cit. a p. 4).

[16] Niyato Dusit et al. «A survey of mobile cloud computing: architecture, applications, and approaches». In: *Wireless Communications and Mobile Computing* 13 (18 dic. 2013), pp. 1587–1611 (cit. a p. 5).

[17] ISTAT. *CITTADINI E NUOVE TECNOLOGIE*. report. ISTAT, dic. 2014. URL: http://www.istat.it/it/archivio/143073 (cit. alle pp. 39, 47).

[18] M. Jelasity e A. Montresor. «Epidemic-style proactive aggregation in large overlay networks». In: *Distributed Computing Systems, 2004. Proceedings. 24th International Conference* (2004), pp. 102–109 (cit. a p. 19).

[19] R. Karp et al. «Randomized rumor spreading». In: *Foundations of Computer Science* (2000), pp. 565–574 (cit. a p. 79).

[20] Jelasity Márk et al. *PeerSim: A peer-to-peer simulator*. Lesson. 2009. URL: http://peersim.sourceforge.net (cit. alle pp. 23, 24).

[21] Jelasity Márk et al. *PeerSim: A peer-to-peer simulator*. Lesson. 2010. URL: http://peersim.sourceforge.net (cit. a p. 23).

[22] Gardner Martin. «The fantastic combinations of John Conway's new solitaire game "Life"». In: *Scientific American* 223 (ott. 1970), pp. 120–123 (cit. a p. 15).

[23] A. Montresor. «Epidemic-style proactive aggregation in large overlay networks». In: *Peer-to-Peer Computing, 2004. Proceedings. Proceedings. Fourth International Conference* (2004), pp. 202–209 (cit. a p. 19).

[24] Marzolla Moreno e Mirandola Raffaela. «Dynamic power management for QoS-aware applications». In: *Sustainable Computing: Informatics and Systems* 3 (4 feb. 2013), pp. 231–248 (cit. alle pp. 4, 5).

[25] Ranganathan Parthasarathy. «Recipe for Efficiency: Principles of PowerAware Computing». In: *Communications of the ACM* 53.4 (apr. 2010), pp. 60–67 (cit. a p. 3).

[26] Hu Ruijing et al. «A fair comparison of gossip algorithms over large-scale random topologies». In: *Reliable Distributed Systems (SRDS)* (nov. 2012), pp. 331–340 (cit. alle pp. 13, 20).

[27] Christian Schindelhauer. *Epidemic Algorithms*. Topic. Paderborn University, 2004. URL: http://www2.cs.uni-paderborn.de/cs/ag-madh/WWW/Teaching/2004SS/AlgInternet/Submissions/09-Epidemic-Algorithms.pdf (cit. alle pp. 15, 79).

APPENDIX

A.1 DIAGRAMMI DI FLUSSO ALGORITMO DYANIMC FANOUT

A.1.1 *Standby FSA*

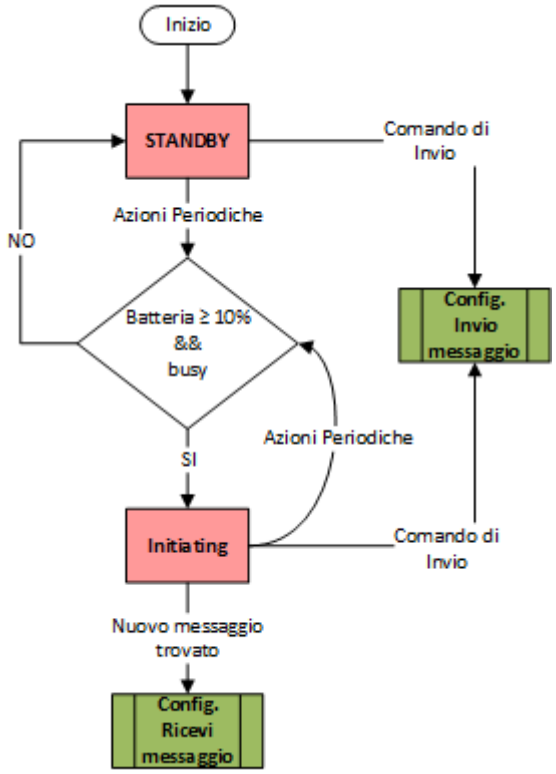


Figura A.1: Diagramma di flusso della macchina a stati Standby.

A.1.2 *Invia Messaggio FSA*

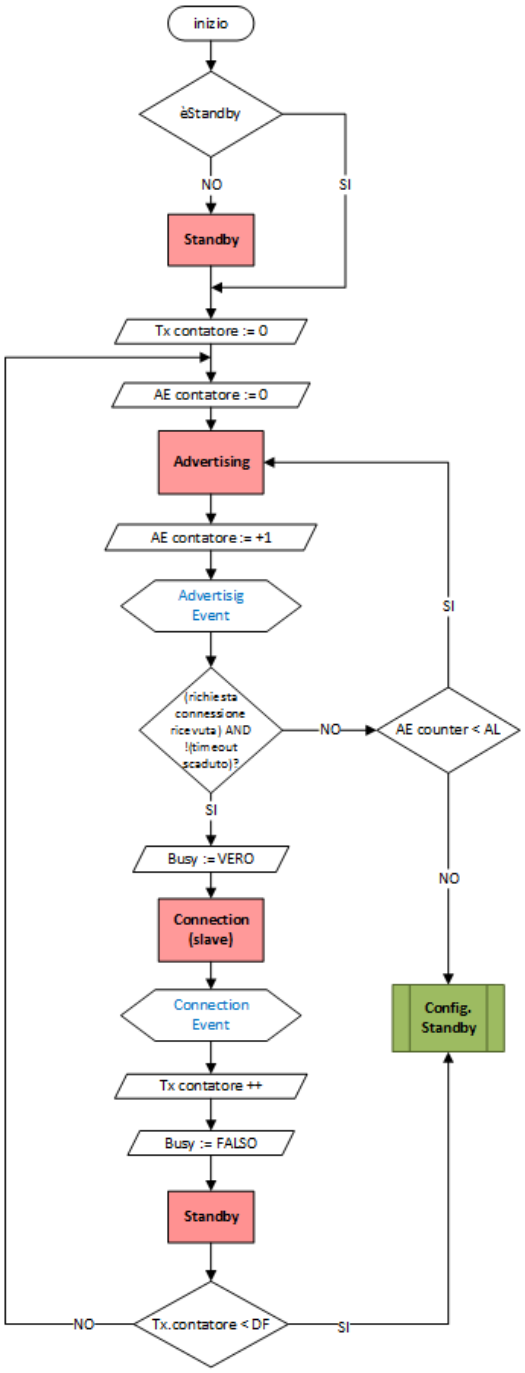


Figura A.2: Diagramma di flusso della macchina a stati Invio Messaggio.

A.1.3 *Ricevi Messaggio FSA*

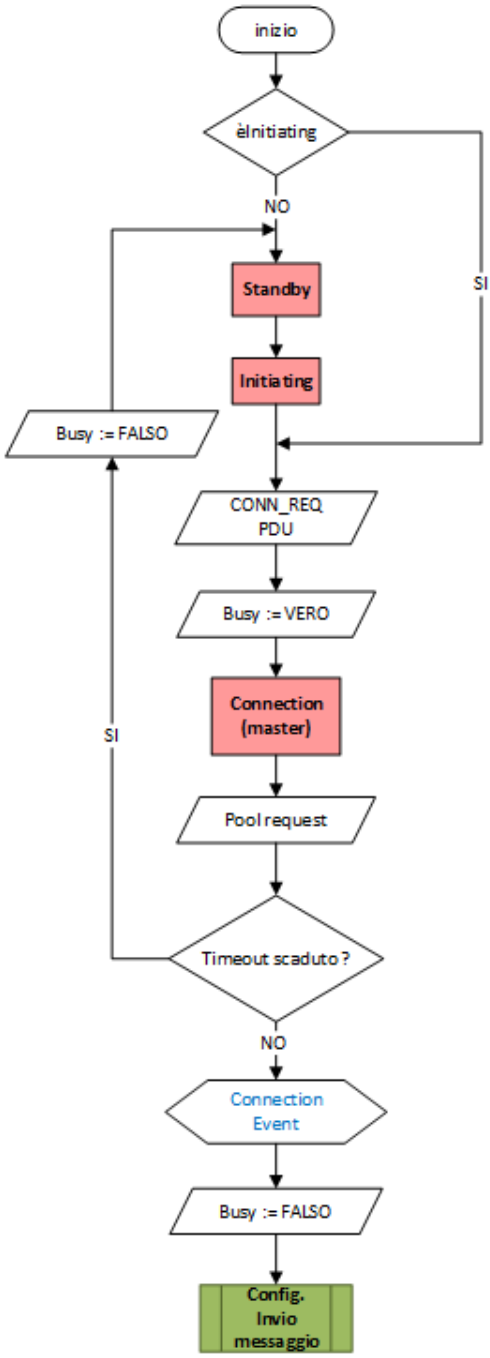


Figura A.3: Diagramma di flusso della macchina a stati Ricevi Messaggio.